



**QLectives – Socially Intelligent Systems for Quality  
Project no. 231200**

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

**Deliverable D4.3.3  
QMedia v3 - Short report**

Submission date: 2012-02-17

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		
<b>PU</b>	Public	<b>X</b>
<b>PP</b>	Restricted to other programme participants (including the Commission Services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission Services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission Services)	



## Document information

### 1.1 Author(s)

Author	Organisation	E-mail
Niels Zeilemaker	TU Delft	n.zeilemaker@tudelft.nl
Boudewijn Schoon	TU Delft	p.b.schoon@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
------	--------------	--------

### 1.3 Document history

Version#	Date	Change
V0.1	December 1, 2011	Starting version, template
V0.5	December 13, 2011	First draft
V0.9	December 16, 2011	Complete first draft, for internal consortium
V1.0	17 February, 2012	Approved version to be submitted to EC

### 1.4 Document data

Keywords	peer-to-peer, distributed mass media, decentralized channels, decentralized wiki
Editor address data	T.Vinko@tudelft.nl
Delivery date	17 February, 2012

### 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

### **Technical University of Delft**

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

### **ETH Zurich**

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

### **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

### **University of Fribourg**

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

### **University of Warsaw**

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

### **Centre National de la Recherche Scientifique, CNRS**

Paris 75006,  
France  
Contact person: Dr. Camille ROTH  
E-mail: camille.roth@polytechnique.edu

### **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

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](http://tribler.org); and the scientific collaboration forum [EconoPhysics](http://EconoPhysics).



# Executive summary

This report accompanies and documents *software* deliverable QMedia version 3.0. This is thus a rather brief technical manual for an Internet-deployed self-organising system.

QMedia is a next-generation social media distribution platform and one of QLectives living labs. Its vision is to provide an alternative to mass media through the concept of self-organising personalised collectives. Version 3.0 of the QMedia software is based on the QLectives Platform version 3.0, described in QLectives Deliverable D.4.1.3. This report concentrates on the changes brought into QMedia since version 2.0, released in Month 24 of QLectives, in particular on the Open2Edit concept, which provides users with a platform to create virtual communities in a completely decentralized setting. By implementing the means to help users monitor each other, as is done in Wikipedia, they themselves have to maintain order in the community. Implemented without any centralized system restricting a peer from remaining in control, we aim to get the same level of quality and activity as Wikipedia.

QMedia version 3.0 has been released in early December 2011. This version of the software, named Tribler 5.5 was downloaded more than 2,200 times since its publication<sup>1</sup>.

---

<sup>1</sup>for more statistics, see <http://statistics.tribler.org>



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Open2Edit</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Related Work . . . . .	4
2.3	Proposed Solution . . . . .	5
2.3.1	Quality . . . . .	5
2.3.2	Activity . . . . .	7
2.4	System Architecture . . . . .	8
2.4.1	GUI Design . . . . .	9
2.4.2	Message Design . . . . .	10
2.4.3	Dispersy . . . . .	10
2.4.4	Dealing with inconsistent views . . . . .	11
2.5	Evaluation . . . . .	12
<b>3</b>	<b>Summary</b>	<b>15</b>



# Chapter 1

## Introduction

This report describes QMedia version 3.0, the third version of an experimental media-sharing distributed software built around the concept of quality collectives. The network of QMedia users serves as a living lab for QLectives, both as a testbed for developed algorithms and providing input for the design and evaluation of new algorithms.

QMedia's vision is to provide a decentralized, user-centric and social media-distribution platform. This platform can provide an alternative to mass media through self-organising personalised collectives. QMedia is based on the QLectives Platform (see D4.1.1 and D4.1.2 for details on the platform), which aims to serve as a generic middleware to develop peer-to-peer (P2P) applications using its central component, a Distributed Permission System, called Dispersy (reported in D4.1.2). The recent development in the QLectives Platform is reported in D4.1.3, here we are focusing on the recent development in QMedia.

In decentralized virtual communities, the need for policies to maintain order is often neglected. Existing solutions often resort to having one central node enforcing the global policy of the community. In this deliverable we use the success of Wikipedia to come up with a new approach for decentralized virtual communities. The Open2Edit part of QMedia version 3.0 provides users with a platform to create virtual communities in a completely decentralized setting. By implementing the means to help users monitor each other, as is done in QScience (see D4.2.3) and Wikipedia, they themselves have to maintain order in the community. Implemented without any centralized system restricting a peer from remaining in control, we aim to get the same level of quality and activity as Wikipedia.

Built using an the QLectives Platform version 3.0, we allow users to publish and consume media files, create and modify metadata, and post comments. Furthermore, in an extensive emulation run on our DAS-4 supercomputer cluster, we show that our implementation is scalable and allows peers in the system to be synchronized with ease. Our main contributions are: (i) identification of key features in Wikipedia used to maintain order; (ii) the design and implementation of a virtual community platform; and finally (iii) performance analysis of the platform.

The new feature of QMedia is also related to WP4.4, where quality metadata needs to be identified and promoted. Open2Edit serves as a spam filter, being a distributed collaborative filtering by its users. Detailed results on testing this feature will be reported in Deliverable D4.4.4 due in month 48.

# Chapter 2

## Open2Edit

### 2.1 Introduction

Decentralizing virtual communities has a big advantage, because it allows users to remain in control of their data. But it also has a big disadvantage: remaining in control while keeping peers from misbehaving is much more complex. This caused many to neglect the policies necessary to maintain order, or led to centralized or impractical solutions [4]. By allowing everyone to modify all pages Wikipedia faces a similar situation, but instead of chaos and mayhem they maintain a high level of *quality* and *activity*. Giles et al. even consider the quality of the articles to rival that of Encyclopedia Britannica [7]. In this paper we transpose some of the core ideas of Wikipedia resulting in Open2Edit, a truly decentralized virtual community platform able to maintain order. Our main contribution provides users with the tools to monitor each-other, resulting in a similar experience as in Wikipedia. Open2Edit allows users to publish and consume media files, create and update metadata, and post comments. Using BitTorrent to distribute the media files, it builds upon proven technologies. Combining Open2Edit with BitTorrent allows users to create a truly decentralized virtual community without the drawbacks of previous solutions.

Wikipedia faces problems similar to a decentralized virtual community due to not imposing any restrictions and trusting everyone. Trusting everyone will make it vulnerable to all kinds of uncooperative behavior, such as vandalism etc. Tapping into the collective wisdom of the crowd both for producing content and for its ability to revert vandalism, Wikipedia seems to have found a solution for

building a virtual community without the use of either restricting policies or a reputation-system. We think that, providing the tools to, but not requiring all users to monitor changes, create, improve and correct articles is the key reason why. In February 2011<sup>1</sup>, Wikipedia had 14,291 million pageviews and 11 million articles were modified by users. A testimony of how well their solution to maintaining order performs, as the quality of the articles remain stable.

## 2.2 Related Work

Being a member of a community has certain advantages; i.e. a member can have access to information which is not available for non-members. A key problem in a community is how to devise and enforce quality. The rules prescribing these are usually referred to as policies. In a decentralized virtual community enforcing policies is the key problem not solved by related work. Pearlman et al. [15] try to solve it by introducing a server in charge of distributing certificates to peers allowed to perform an action. Using these certificates, peers can perform actions in the network by handing in these certificates as proof. Boella et al. [4] extend this work by differentiating between global and local policies. They state that authorizing a peer to perform an action is not the same as permitting it. A peer can still locally decide whether or not to comply. We think that an actual implementation, which allows for local and global policies would be very complex. As all peers can locally decide not to comply, a solution should be found which forces them to comply, basically stripping them of the added control they have over their data due to decentralizing.

Another approach could be to only cooperate with trusted peers. Peers who behaved nicely in the past could be trusted more than those who did not. As Abdul-Rahman et al. [1] state “At any given time, the stability of a community depends on the right balance of trust and distrust”. By building a distributed reputation network, a peer could build a graph using his locally trusted peers. Such a reputation-graph could then be used to see if a peer, being part of the same community, can be granted access to a local resource. Fully distributed reputation systems have been developed [10, 14, 16], but suffer from basic attacks which

---

<sup>1</sup>Sources: <http://stats.wikimedia.org/EN/TablesPageViewsMonthly.htm>,  
[http://stats.wikimedia.org/reportcard/RC\\_2011\\_02\\_detailed.html](http://stats.wikimedia.org/reportcard/RC_2011_02_detailed.html)

render them useless. An example of such an attack is whitewashing, in which a peer after receiving a low reputation simply creates a new identity and abuses the “free” trust given to new peers. Countering this simple attack has proven very difficult and relies on making the creation of new identities more expensive [5].

## 2.3 Proposed Solution

In this section we will elaborate on our solution. Open2Edit is transposing some key ideas from Wikipedia in order to create a fully decentralized virtual community platform. Allowing our users to create virtual communities sharing media files, create and update metadata, and post comments we intend copy the success of Wikipedia in a decentralized setting. Combined with BitTorrent Open2Edit allows users to create a truly decentralized virtual community without the drawbacks of previous solutions. Our approach defines two key aspects which we want to promote, *quality* and *activity*. Combined, these two aspects are what we think key to the success of Wikipedia. Quality defines the accuracy, completeness and resilience to vandalism of articles. Promoting it will improved the value of a community. Promoting activity, will cause the sheer number of contributions to increase. By itself promoting activity does not increase the value of the communities, but when combined with promoting quality it will. Due to users being rewarded for good and penalized for bad behavior.

### 2.3.1 Quality

Quality of articles in Wikipedia is maintained by users and bots (automated scripts which monitor articles). Together they revert acts of vandalism when needed and warn/ban a user if necessary. Friedhorsky et al. [17] state that 42% of damage to articles is reverted almost immediately, a testimony of how well the system works. Wikipedia provides users with ‘Recent changes’ lists, which show all changes sorted by time. A small sample of frequent Wikipedia users are actively monitoring those pages to prevent vandalism making an impact [18]. Furthermore, users can be notified to changes of a particular article, for instance articles they wrote themselves. Wikipedia’s NPOV (Neutral Point of View) principle states that an article should not reflect a the view of a single user, but that of the community as a whole. To allow for this, it uses talk-pages to separate discus-

sion from the article. Having a place to share thoughts will result in less changing articles and improve quality.

Open2Edit provides the above functionality in a decentralized setting. Similar to Wikipedia, users can monitor each other using public lists where all changes are posted to. Warnings are published on public lists, visible to all users. Changes made by warned users are flagged and placed higher on the recent changes lists, allowing for faster detection of vandalism. This behavior is similar to that of Huggle [6], a tool which is used to revert vandalism in Wikipedia. Huggle is constantly monitoring the recent changes list of Wikipedia and will order those changes based on a number of heuristics. For instance, if a user has received multiple warnings, Huggle places an icon in front of every change made by this user. Open2Edit replicates this and additionally shows a small warning on any item modified by this user.

*Revision History*, comparable to “page history” in Wikipedia, lists all modifications. Visible for a single item, category or community, the list has two distinct modes. One shows all changes, the other shows the changes related to a user. These lists, in their personalized mode, can be compared to the watchlists of Wikipedia which allow users to monitor their favorite articles. As stated by Viegas et al. [18], a small sample of frequent Wikipedia users are using these to look for unknown/newcomers which are potentially a bigger vandalism threat due to not knowing community standards. First-time contributors are highlighted in the list, as are warned users albeit differently.

Every item, category and community will have a *comments page*. As described in the previously, these will improve the stability of the community as a whole by allowing users to have discussions separate from the actual items. To lower the effort required to thank an uploader we will implement “thank you” buttons. These will thank the user who made the initial upload, with only a marginal effort required. Users of Slashdot who posted their first comment are less likely to post another one if this post is not rated [12]. We think that the “thank you” button will increase the number of replies to an upload, improving the willingness of users to contribute another item.

Similar to Wikipedia, any user can become a moderator, but has to be granted the rights to do so. Having more than one moderator will allow for more stable communities requiring less effort from a single moderator. A user creating a com-

munity can transfer his rights to others and leave the system completely without the community having to die as well.

### 2.3.2 Activity

By harnessing the collective wisdom of the crowd, we are promoting activity. As our solution to improving quality depends highly on users, we think that the effort required to make a contribution/improve the community should be kept as low as possible. We will now show *three* examples indicating that if the effort necessary for one contribution is kept very low, then many more users will contribute, thus increasing the overall activity in a community.

As a first example of how a low effort task can boost contributions; NASA [11] used Internet volunteers to identify craters on mars. Users were first trained using craters which were identified previously, allowing for feedback and improving their skills. A spike in interest caused 30 000 craters to be identified in four days, most by dedicated/returning users but 37% of all craters were identified by users which only visited the site once. Kanefsky et al. state that “Even with redundancy for cross-checking, this is faster than a single graduate student could have marked them, and also far faster than the original data was returned by the spacecraft.” a testimony of how powerful crowd-sourcing can be.

Another examples is Wikipedia, which provides users with an edit button allowing them to modify each article. Being bold, a statement often referred to in Wikipedia[2], emphasizes that the encyclopedia is built by users for users and that everyone can/should edit an article if they think it is lacking. Bots are looking for possible errors, such as missing references, and highlight these to encourage users to modify/fix them.

Another effort of building an encyclopedia using Internet volunteers shows that requiring more effort will cause collaboration to suffer dramatically. In Google Knol users first have create an account, then suggest a change and wait for the owner of an article to accept it (as described in [8]). Popularity of Google Knol is decreasing and some have questioned if it has a future at all, stating that its biggest problems is that “knol-body is reading” it.<sup>2</sup>

In Open2Edit we aim to increase activity level by lowering the effort necessary

---

<sup>2</sup>See the original full article at: <http://arstechnica.com/web/news/2009/01/google-knol-six-months-later-wikipedia-need-not-worry.ars>

for each contribution. We expect to improve the quality of meta-data by allowing all users to edit/improve them. I.e allowing users to add descriptions, allowing them to assign categories etc. Furthermore, we promote users to start monitoring items they have shown interest in by creating a personal view of the recent changes and activity lists. Finally, users are suggested actions after completing a download, such as adding meta-data, thanking the original uploader etc. We call this feature *wiki-like editing*. Allowing us to tap into the vast resource of users and improve the overall value of the communities.

Furthermore, we implemented features which allow for greater *differentiation* between users and communities (as suggested by Hummel et al. [9]). A community can distinguish itself from other communities by having a more appealing look, which helps to set it apart. Inside a community, users will have their own page, accessible from every contribution made by a user, which gives an overview of all his activities. Similar to Wikipedia, a userpage has comments which can be used as a direct method of communication to him.

Finally, we are introducing a greater *difference of being or not-being a member* of a community. By hiding most content of a community when a user is not a member, even if an user has more content locally available, users are lured in. Then after joining a community, a user will have access to all content. To prevent all users from simply joining all communities, we require a minimum effort from all. Increasing the activity levels and the overall community feel. This effort is currently implemented as simply requiring a user to post a message within a 1 month period (i.e. he/she could comment on one item).

## 2.4 System Architecture

Open2Edit is composed of three parts, the *GUI and message design*, *Dispersy* our distributed permission system in the QLectives Platform, and the BitTorrent engine of QMedia used to exchange media files. In this section we will first describe in detail the GUI, then message design, and we give a short overview of Dispersy (for full details, the reader is referred to D4.1.3).

## 2.4.1 GUI Design

Most features as described in the previous sections are visible in the GUI of Open2Edit. Amongst these features are allowing users to add their own items, modifying and categorizing them. Commenting on items, categories and communities etc. When designing the GUI, effort was put into making sure that the position of controls and easy of use was such that it promoted their use. Similar to the 'Be Bold' statement which Wikipedia used to persuade users to start modifying and improving the encyclopedia.

Using the activity list, a user can revert to a previous version after specifying a reason. Templates are available which allow for default responses. Templates help to improve learnability of the system, where a user is educated on what the norm is within a community. A clear difference is made between also issuing a warning, or just reverting. Slashdot uses a templates for moderator ratings. A user is up for a possible banning after receiving 4 warnings, and similar to Wikipedia this can only be done by the moderators. And a warning will expire after a set period.

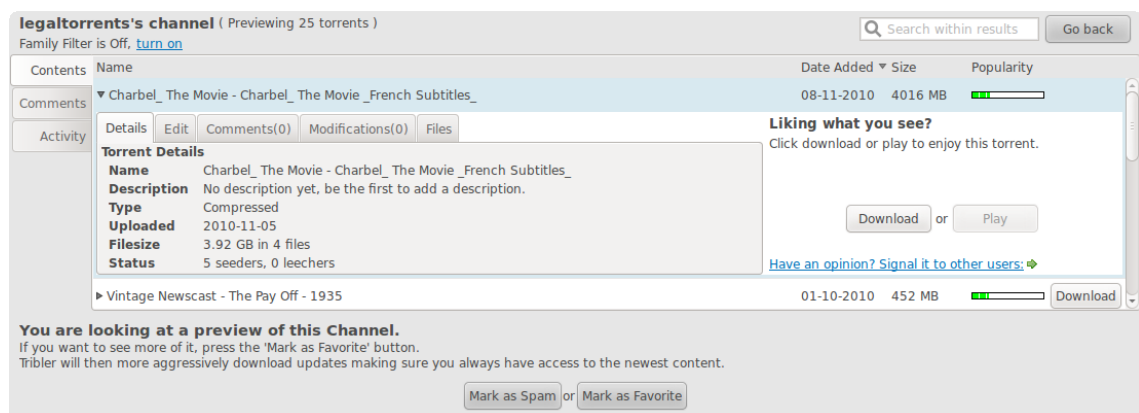


Figure 2.1: An overview of the Open2Edit GUI

Figure 2.1 gives an overview of what is implemented in Open2Edit. This is a screenshot of our implementation, ready to be deployed. It shows a community in its preview state, a community this user did not join yet. In the community an item is expanded with the edit, modification and comments tabs. On the left are the 'Activity' and 'Comments' tabs for the community as a whole. The 'Activity' tab is a list containing recent activity, such as new items, modifications or comments, warnings issued etc. What is not shown are controls for the warning and

banning, these are only visible from the activity and modification tabs.

## 2.4.2 Message Design

We are defining two different Dispersy communities. Defining a Dispersy community basically configures it; i.e. which messages are going to be used, who can send them etc. In Open2Edit we are defining two communities; the *AllChannelCommunity* designed for discovering new communities, and the *ChannelCommunity* which is used when users have joined a community. There is only one instance of the *AllChannelCommunity*, which all peers join, but a new instance of the *ChannelCommunity* is created for every community created by a user.

In the *AllChannelCommunity*, a *ChannelCast* message is sent by a peer to one other peer every 15 seconds. This message consists of 55 items (only the identifiers are sent), 25 are selected from a user's own community promoting it to other users. 25 are selected from communities this user has marked as a favorite and 5 from communities this user has downloaded at least one item from. Upon receiving a *ChannelCast* message, a peer will request all items he is interested in. Usually a peer will request all unknown items, with the exception that if a user has marked a community as spam then no items from this community are requested. After discovering and joining a community, a peer will additionally send its vote (as a signed *VoteMessage*) to the *AllChannelCommunity* signaling this to other peers. Furthermore, a peer will start synchronizing all data available for it. Starting with the latest, as we expect that this data is most popular. By using the *VoteMessages* a peer has received from others, a peer will customize its *ChannelCast* message by not sending any duplicate items, as peers who have joined a community will already have most of the data. As the *ChannelCast* message only has 55 slots, this additionally will improve the discovery of community a user has not joined.

## 2.4.3 Dispersy

In Open2Edit we have three basic operating modes defining *user management*. A community can be configured as being completely closed, resulting in the moderators injecting all content, being the only ones to modify meta-data, and disabling comments. The second mode, will allow for comments, but still changes

can only be made by the moderators. The third mode, resembles Wikipedia the most and allows all users to modify meta-data, inject new items and categorize them. Moderators in this mode will still be the only ones which can delete items, but users can suggest items which should be deleted and the GUI will modify the appearance of items which are flagged for deletion.

Required by this design of Open2Edit is a permission system. As moderators have more rights than normal users, some messages need to be designed such that only the moderators of a community can send them. Dispersy, which has been developed in the QLectives Platform, provides a fully decentralized permission system, which can be used to grant users more rights to become a moderator etc. Furthermore, it provides a reusable flexible infrastructure in which we can define (besides the actual payload) for each message the authentication, distribution, destination and resolution policies. These policies define if and how a message should be signed. The method used to distribute a message, to which peers it should be distributed. And finally, if and how permissions should be handled.

Dispersy uses Bloom filters to detect which messages still need to be synced. Bloom filters [3] are a compact representation of a set of items. It uses a combination of  $k$  hash functions to set  $k$  bits in a bit string to 1. After receiving a sync message, a peer will reply with messages for this community, which are not present in the Bloom filter it received. Using Bloom filters, we can guarantee that no duplicate messages are sent to a peer.

#### 2.4.4 Dealing with inconsistent views

Open2Edit, being implemented as a distributed virtual community platform, has to deal with conflicts. Conflicts could potentially be hindering the collaboration due to users being annoyed because their contributions are being overwritten by not yet received messages. Conflicts happen when two users modify an item roughly at the same time. To detect this, modifications are implemented as a linked-list with each message having reference to the message it is replacing. A conflict can then be detected by seeing if a message is referenced more than once, allowing us to warn users of an existing conflict. Any user can merge two conflicting modifications, but a conflict is only shown in the user interface if the most recent modification has a conflict. Conflicts are additionally shown in the activity

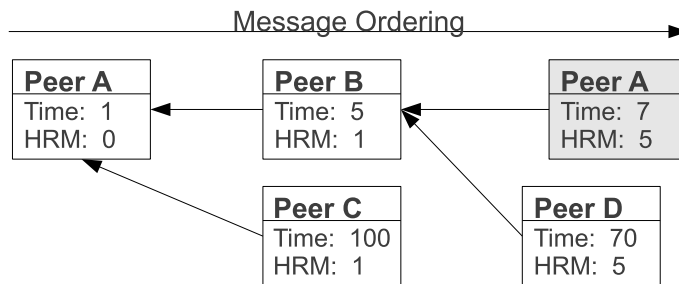


Figure 2.2: Dealing with conflicts, all messages ordered by HRM. Message in grey is shown in the GUI as it has the highest HRM and lowest peerid.

list, which in its personalised mode can notify a user which created the modification of a conflict.

Dispersy implements a Lamport logical clock [13] to order messages. Implemented as a property called time, it is the highest time received by a peer + 1 at message creation. This time is not unique, but indicates for an ordering of messages received from a single peer. Using time, we implemented a linked-list using the highest received message (HRM) of the same type and in the same category at message creation.

An example; when a user has modifies an item, i.e. changing the name of it, he will create a Modification message. This message will have a link to the torrent it is modifying and a link to the highest received modification (HRM). If we assume that the quality of a modification which it is replacing is higher, then sorting the modification messages based on their HRM will result in the best ordering. Because assuming that the quality of modifications is ever increasing is not always correct, the GUI shows a user that his or her modification has been replaced due to this fact. Furthermore, there still is a situation when we need to resolve a conflict, due to two messages having the same HRM, we then resort to sorting the messages by unique peerid. A full example is shown in Figure 2.2.

## 2.5 Evaluation

To evaluate the performance of our system we conducted one experiment on the DAS-4 supercomputer cluster<sup>3</sup>. For this experiment we used a trace of an actual community. This community is currently using our previously deployed system

<sup>3</sup><http://www.cs.vu.nl/das4/>

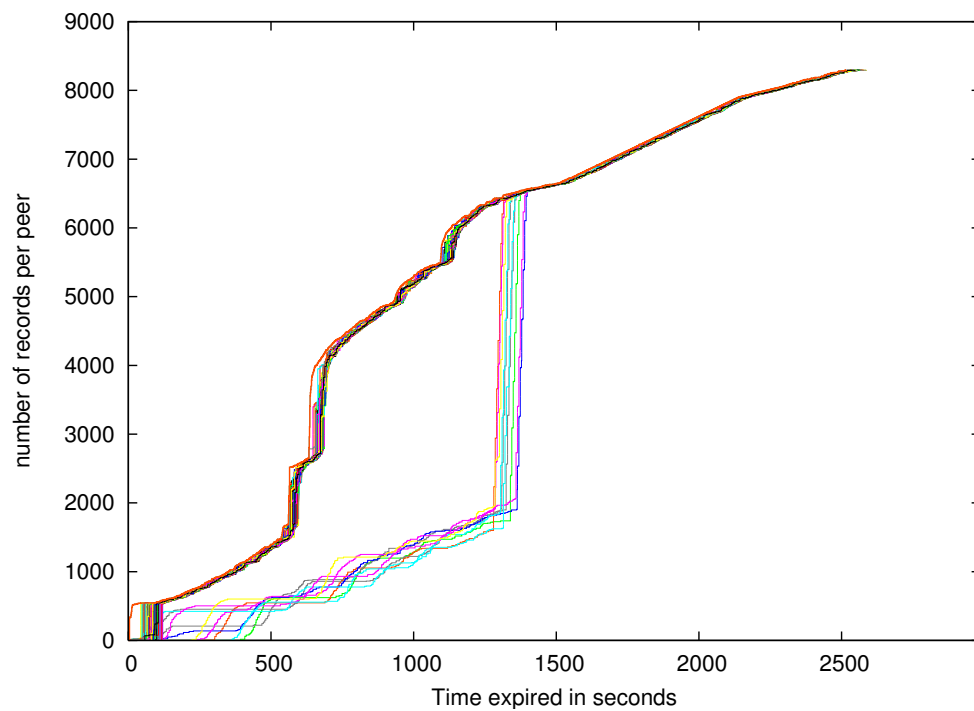


Figure 2.3: Real-life workload of a community: total number of messages at a peer. A line or dot represents a single peer.

to exchange media-files. Its workload was recorded using a crawler, resulting in a daily activity log of the community. The community has over 8,000 items which have been inserted over a period of 6 months. We then emulated this community, but sped up the experiment such that 1 second equals 2 hours in real life. For the experiment we used 10 nodes from the DAS-4, each running 25 peers. The experiment ran for 50 minutes.

The results are shown in Figures 2.3 and 2.4. In our experiment there is one peer, the community owner, which is injecting all items. The other peers join the community randomly within the first 100 seconds. Some peers are configured to start joining after timestep 1300, indicating the maximum catchup speed. Peers can only join if they have discovered the community using the AllChannelCommunity and ChannelCast messages as described in Section 2.4.2.

We want to highlight three things shown in the figures; *first* all peers which have joined the community can easily stay synchronized with the community owner. Except when the owner is injecting large amounts of items, most peers have all data. Second, at timestep 1300 when the remaining peers join they catch up very quickly. As an example, peer 248 joined at timestep 1338 and caught up

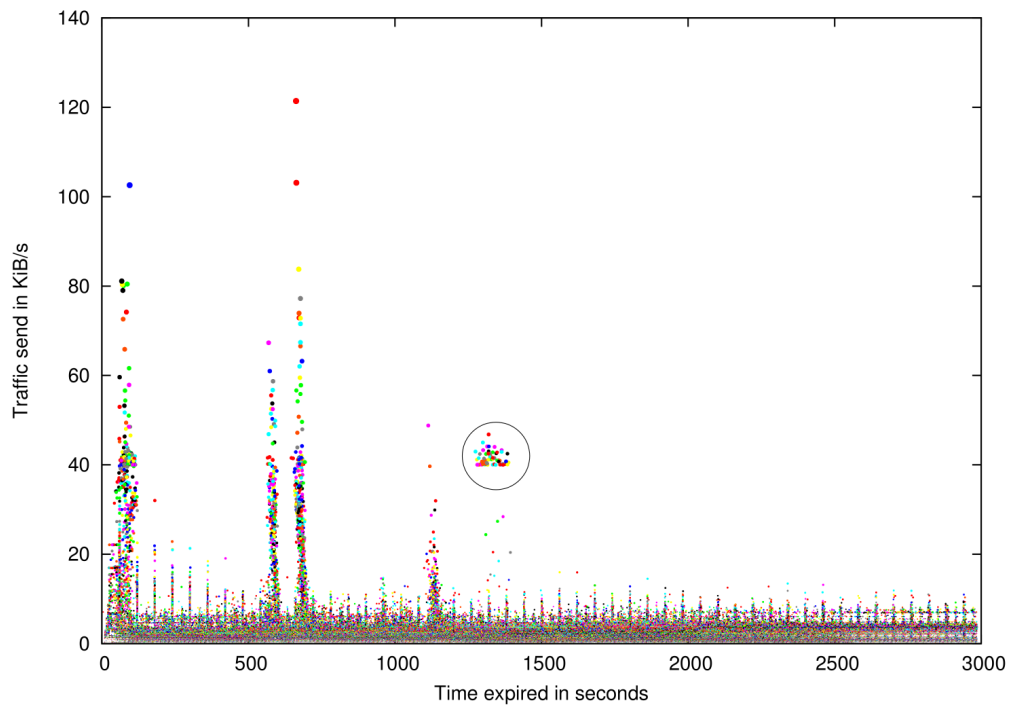


Figure 2.4: Real-life workload of a community: Bandwidth used sending messages. A line or dot represents a single peer.

at timestep 1374, a period of just 36 seconds, in which it sent 7 sync messages and received 4761 items. This translates to roughly receiving 680 items for every sync message sent. Third, when the peers join at timestep 1300, the load of sending them their missing messages is evenly divided. This can be seen in Figure 2.4 where a cluster of peers at timestep 1300 sending 40KiB/s is clearly visible. The total amount of data sent during the experiment is 890.60 MiBytes or 3.5 MiBytes per peer. Roughly 66% is used by Dispersy and only 33% is used by Open2Edit. The sync messages are causing most traffic, every peer will send 2 of these every 5 seconds. If we increase the sync interval of the AllChannelCommunity, the bandwidth used by sync messages will go down. This will cause VoteMessages to be spread slower, but we expect more change within a channel compared to votes.

# Chapter 3

## Summary

This report documents the third release of QMedia, a media-sharing P2P system built on the QLectives Platform, describing the main improvement upon the previous version, the concept and implementation of Open2Edit. The current version of the software is already deployed so as to serve as the QMedia Living Lab.

With Open2Edit we have implemented and evaluated a fully decentralized virtual community platform, including a well-designed user interface. Our approach, using features transposed from Wikipedia, implements a solution to maintaining order in a decentralized setting. Not implementing a reputation system, was a clear design choice and allowed us to implement additionally features focused on improving activity. Open2Edit was designed such that the effort required for contributing in our system is low, simulating collaboration.

Furthermore, due to using a distributed permission system, Open2Edit provides a flexible solution which can be applied to different types of virtual communities. Warning and banning users is completely public and can be verified by all. To allow users to monitor each other activity lists are implemented, which can additionally be personalised. Detecting the interest of a user by looking at its comments, downloads and modifications will allow us to build such lists.

Finally, Open2Edit has implemented features which help resolve problems caused by its distributed nature. If conflicts occur, this is shown to the users, allowing them to resolve it. Detecting conflicts is important as we do not want users which have made modifications based on old data to overwrite a already improved one.



# Bibliography

- [1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 00(c):9, 2000.
- [2] Angela Beesley, Elisabeth Bauer, and Kizu Naoko. How and Why Wikipedia Works : An Interview. *Computers and Society*, pages 3–8, 2006.
- [3] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.
- [4] Guido Boella and Leendert Van Der Torre. Permission and Authorization in Policies for Virtual Communities of Agents. *Informatica*.
- [5] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 24(5):1010–1019, May 2006.
- [6] R Stuart Geiger. The Work of Sustaining Order in Wikipedia : The Banning of a Vandal. *Administrator*, pages 117–126, 2010.
- [7] Jim Giles. Internet encyclopaedias go head to head. *Nature*, 438(7070):900–1, December 2005.
- [8] Felix Halim, Wu Yongzheng, and Roland Yap. Wiki credibility enhancement. *Proceedings of the 5th International Symposium on Wikis and Open Collaboration - WikiSym '09*, page 1, 2009.
- [9] Johannes Hummel and Ulrike Lechner. Social Profiles of Virtual Communities. *Dimension Contemporary German Arts And Letters*, 00(c):1–10, 2002.
- [10] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. *Proceedings of the twelfth international conference on World Wide Web - WWW '03*, page 640, 2003.
- [11] B Kanefsky, NG Barlow, and VC Gulick. Can Distributed Volunteers Accomplish Massive Data Analysis Tasks? In *Lunar and Planetary Institute Science Conference Abstracts*, volume 32, page 1272, 2001.

- [12] Cliff Lampe, Paul Resnick, West Hall, and Ann Arbor. Slash ( dot ) and Burn : Distributed Moderation in a Large Online Conversation Space. *Computer*, pages 1–8, 2004.
- [13] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.
- [14] M Meulpolder, J Pouwelse, D Epema, and H Sips. Bartercast: Fully distributed sharing-ratio enforcement in bittorrent. *Delft University of Technology-Parallel and Distributed Systems Report Series*, 2008.
- [15] Laura Pearlman and Carl Kesselman. A Community Authorization Service for Group Collaboration The University of Southern California Department of Computer Science The University of Chicago Division , Argonne National Laboratory The University of Chicago The University of Southern California. *Auditing*.
- [16] Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, and Thomas Anderson. One hop reputations for peer to peer file sharing workloads. pages 1–14, April 2008.
- [17] Reid Priedhorsky, Jilin Chen, Shyong (Tony) K. Lam, Katherine Panciera, Loren Terveen, and John Riedl. Creating, destroying, and restoring value in wikipedia. *Proceedings of the 2007 international ACM conference on Conference on supporting group work - GROUP '07*, page 259, 2007.
- [18] Fernanda Viegas, Martin Wattenberg, Jesse Kriss, and Frank Ham. Talk Before You Type: Coordination in Wikipedia. *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pages 78–78, January 2007.