UNIVERSITY OF TARTU

Institute of Computer Science

Software Engineering Curriculum

Ioane Sharvadze

Implementation of WebWeaver Platform over P2P Network

Master’s Thesis (30 ECTS)

Supervisor(s): Vishwajeet Pattanaik, PhD

Tartu 2018

**Title**

**Abstract:**

In the era of Web 2.0 we are using numerous applications that store and manage user data. Switching one application with competitor seems a problem, since we are losing data. While there are numerous approaches to this problem, we propose P2P solution, that gives users real ownership of the data. The goal of this thesis is to show how we enable to have similar functionality of centralized database while storing data on user’s devices.

This thesis will show solution for this particular application WebWeaver, that enables users to add a comment on a web page and share with other users, without website having this feature.

Keywords:

Social blogging, Liquid democracy, Digital divide, Peer-to-peer network

Table of Contents

[1 Introduction 4](#_Toc504859272)

[1.1 Goals 4](#_Toc504859273)

[1.2 Importance 4](#_Toc504859274)

[1.3 Requirements 5](#_Toc504859275)

[1.3.1 Private Share 5](#_Toc504859276)

[1.3.2 Public Share 5](#_Toc504859277)

[1.3.3 Private Shared 6](#_Toc504859278)

[1.3.4 Multiple Devices 6](#_Toc504859279)

[1.3.5 Social Sharing 6](#_Toc504859280)

[1.3.6 Edits, Deletes 6](#_Toc504859281)

[1.3.7 Security, Integrity 6](#_Toc504859282)

[1.3.8 Technical 6](#_Toc504859283)

[1.4 Comparison & Overview 6](#_Toc504859284)

[1.4.1 Musubi 6](#_Toc504859285)

[1.4.2 Linked Data, Solid 7](#_Toc504859286)

[1.5 Limitations 7](#_Toc504859287)

[2 Proposal 8](#_Toc504859288)

[2.1 Why Server? 8](#_Toc504859289)

[3 Implementation 9](#_Toc504859290)

[4 Outcome & Conclusion 10](#_Toc504859291)

[5 References 11](#_Toc504859292)

[Appendix 12](#_Toc504859293)

[I. License 13](#_Toc504859294)

# 1 Introduction

## Goals

This thesis is about enabling Peer to Peer data sharing between users. The goal is to remove centralized server as data store and empower users with real data ownership. Ultimately, we would like to remove server at all and make users communicate with each other without any middleware, but later thesis will show why this is not possible.

This thesis will concentrate building solution for a specific application, called WebWeaver. This is a tool that enables users to weave data into specific web element without knowledge and support of the website. WebWeaver is currently being developed as Chrome Extension, that adds this feature to all websites. This thesis will create simple library, that enables application to store data on user’s device and directly share it with specific users. For establishing communication between users, server is needed, but actual communication and data transfer happens on Peer to Peer network.

## Importance

This section will explain the importance of this research.

There are two major problems associated with centralized data management. Data Ownership and privacy. Whenever using of any well-known applications such as Facebook, Google Plus, Twitter etc. we see that they own and manage our data and it is impossible to continue using other application without losing account data [1]. Imagine after years of using Facebook, user wanted to use another application, that would have similar features, but different interface, or provide extended solution. User would not be able to transfer its data to the new website. Data privacy is also concern for users. While people send messages to each other, their data is stored on third party servers.

While there are different proposals to tackle this problem, yet none of them are widely adopted [1].

While area of decentralized data management remains in active research, topic should be considered as important.

## Requirements

As stated before, this thesis will build this solution for a particular application, called WebWeaver platform. To model adequate solution, we need exact requirements that WebWeaver platform [2] has stated for data management layer. For that we need to understand what application does.

WebWeaver is an annotation tool that enables users to leave a comment on a web comment. They can mark comment as public, private or share it to specific users. WebWeaver uses robust algorithm to detect anchors on web pages [3]. Unlike other web annotators like *hipothes.is* [4] or *genius* [5] WebWeaver aims to improve anchoring strategies in case of dynamic content.

To explain the anchoring problem, let’s describe dynamic page like Facebook. Once I enter the page I will see a different content then the user with different account. In this case if I annotate any element, the other user (or even me on next day) might now see same element at all. This is still an open issue for annotators, because of its complexity. But WebWeaver tries to solve it using similarity calculation [3].

This thesis goal is to provide communication mechanism for WebWeaver, so anchoring algorithms will not be discussed here. But still as for requirement proposed database solution will have to transfer data of a small size, as for anchoring algorithm metadata is relatively small and average comment will require less then 10 kilobytes in size. That means that data sharding and parallel downloads like BitTorrent [6] protocol does, will not be needed, as for parallel downloads matter only when data to be transferred is large enough, because establishing P2P connection itself is slow [7] due to NAT traversal technics.

WebWeaver is a social application, that means that it can share data to other users, make it public or leave it as private. That means security and privacy should be considered.

### 1.3.1 Private Share

Imagine Michael added a private annotation on a web page on his one computer. In this case, only Michael should be able to see this annotation.

### 1.3.2 Public Share

Michael added a public annotation on a web page, that means that everybody in the network should see the annotation when entering same web page.

### 1.3.3 Private Shared

This is the scenario when Michael added annotation and wants to share with his friends. In this case, only friends should be able to see annotation.

### 1.3.4 Multiple Devices

Note that people might have different devices. So, in the first case when Michael added annotation, his other devices with same account should access annotations.

That means that solution in this thesis should handle multi device accounts.

### 1.3.5 Social Sharing

Another Requirement of the application is that, if Michael publicly published annotation and now other users have seen annotation, new user might download annotation from another user.

### 1.3.6 Edits, Deletes

This is one of the hardest case to enable editing and deleting items. After feed is distributed in network and author decides to edit or delete, it should be updated for other users too.

### 1.3.7 Security, Integrity

While network is distributed, it should be possible to validate by another user that annotation that was downloaded is unequivocally created by the specified author. Also, private annotation should not be accessible from other users.

### 1.3.8 Technical

Technical requirements are that, library should work well with Chrome browser and should be able to be integrated in Chrome Extension, as far as WebWeaver is a Chrome Extension.

## Comparison & Overview

This section will overview different approaches for decentralized data management and will show in what sense is our approach better.

### 1.4.1 Musubi

Mobile social application platform Musubi [8] proposes Trusted Group Communication Protocol to send encrypted data with server relay. This is very mobile application based solution that uses server to transfer data, because some of the mobile networks (3G), do not allow incoming connections [8].

Musubi users can send their data only after public key is shared among other users. Its goal is to provide a Framework to develop server less mobile apps. While it’s goals seem to be like ours, it is restricted to group sharing, while in our requirements must support public data sharing, so that people who do not know each other and are not friends can still share some data.

### 1.4.2 Linked Data, Solid

Solid is very interesting platform that was built specifically to target data ownership. It uses Web ID to identify users. User should choose its own data server. Service can be third party or self-hosted, but it should implement Solid interface to support all the features. It uses RDF-based resources to link data [1].

The reason why we don’t want to go with Solid is that it still stores data on non-user device. It can be hard for user to set up service, or find any free server. So, it could be better if without any configuration, user could start using application, but also have own data on personal computer.

## Limitations

Research showed that it is impossible to establish Peer-to-peer communication between users without third party server [7]. The reason is that in real environment, most of devices are hidden behind Network Address translators (i.e. NAT). They give user a temporary IP for communication, they only allow traffic from IP’s that user has requested information.

That means that if user has open IP on its device, NAT will not give any public IP, or even if it gives, it won’t allow incoming connection requests.

To overcome this problem, peers should start requesting connections to each other simultaneously. In that case NATs will most likely (64% times for TCP connections) enable Peer-to-Peer connection. This technique is called Hole Punching [7].

Unfortunately, that means for us that in some cases it is not even possible to send data to peer without server. That case should be always considered and as a fall-back mechanism, relay should be allowed. That means that if Michael wants to send a data to Dwight, both peers will need to connect to the public server (that obviously is not behind the NAT) and first Michael will send data to server and server will redirect the data to Dwight.

# Proposal

In this section, I propose solution, while in next section, implementation will be started based on this proposal.

## Why Server?

Let’s revise limitation, as first research revealed that it is impossible to implement server less application, that will connect users and make it possible to transfer data. There are several reasons:

### P2P connection establishment

To make a peer to peer connection we need a signalling server, because peers must know when another peer wants to connect. Before establishing connection, client data should be sync, ports opened, and metadata transferred. This topic has been long researched that’s why Google standardized peer to peer connection and created WebRTC (web real time communications) protocol [9]. WebRTC is a browser-based API that is implemented by most of the contemporary browsers. It has been implemented first in Google Chrome browser and thus it fulfils our requirement. WebRTC is just an API, for implementation various signalling implementation should be used. For the sake of implementation, I tend to use *Socket.io* [10]that is a simple library to implement it on our server.

### Public data holding

In order to share public data, it needs to be stored somewhere on the server, otherwise for example if user requests annotations for website it has to ask to all users. It’s obvious that asking all users is not a feasible solution.

### Sharing data when Offline

Another reason for using server side application is that users might not be online when data is shared. Imagine scenario when Alice shares something with Bob, but Bob is not online. In this case when Bob comes online, Alice might not be online. Data will not be shared unless both are online. If we have server this scenario will be easily fixed. Alice will leave a message to Bob, when Bob gets online he will check message that is stored on server.

## Server Architecture

As far as we need a server, I want to make it application agnostic. That means that server is very generic, that can serve any type of application, without any registration needed. Server is not trusted and it should not know anything about the data that is shared. Also, application should be ready to work in case server fails and deletes the data.

That idea is very similar in Musubi [8] that enables developers to develop application without server code. My goal is same as Musubi’s, except it will not be a mobile first approach and also will enable users to make peer to peer connection.

Thus Server has two services, “Message Box” and “Live Room”. These are two distinct modules that enable application developers to implement any kind of data sharing application.

First let’s discuss Message Box, that is a simple place where message can be left. It stores those messages temporarily, once message is read it cleans up information. Let’s say Alice wants to leave a message to Bob, when Bob reads a message, “Message Box” deletes a message, because Bob already received a message. Also, Alice can decide to delete message that has not been read yet. If Bob did not read a message for a long time (he was out of the reach and did not check the message box) message will be deleted without delivering.

“Live Room” is much easier concept. Every user who is online is waiting for incoming connection, imagine Alice is online and Bob is online too, they do not have to leave a message in a Message Box, but rather contact directly.

Users of our server will not have to register, but they will have their public and private key. User management and how users can add friends will be discussed later in this thesis, but for now we can imagine that users have public and private key and users know public keys of their friends.

## Message Box

The first step of implementing server is a message box. As explained above it’s a simple database where it holds messages. Message is deleted after it’s read, or certain time has been passed. Imagine a situation where Alice shares a message, but Bob does not use this app quite some time. In this case, there is no reason to hold this information anymore, because this message will not be read soon.

Server should not be trusted, so data should be always be encrypted and checked for modifications. For that reason, application (user side) will always deliver messages as encrypted so that only the recipient will be able to read a message, even if Message Box broadcasts this message to adversaries nobody will be able to read the content. This is a good practice, because users have most of the power and they do not trust and rely on anybody except themselves.

Message Box Should Also support public messages, that should stay longer if they are popular.

## Live Room

As Explained before, Live Room is a place where data can be sent P2P. Live Room would handle all cases if all the users were online all the time. But because in real live people are not always on, we need to extend service with Message Box.

The idea of Live Room is to enable fast (real-time) data sharing between users with P2P traffic. Some applications might prefer to only user Live Room, because no server will ever hold the data. But still data encryption is absolutely required, as for by nature of P2P connection establishment [7], data can be easily leaked.

As for Live Room, it will be place where online users gather and wait for incoming connections. Each user will connect to its own “Room” that will be associated with user id. Whenever anybody wants to connect with user, they will connect in this room and ask for connection.

After asking, user will decide whereas they want to connect or not. Some applications might choose to block connections from unknown users, while some might allow. Connection blocking will be decision for application.

After users are connected, they can send data, verify delivery and close connections.

# Implementation

# Outcome & Conclusion

[9]

# References

|  |  |
| --- | --- |
| [1] | E. Mansour, A. V. Sambra, S. Hawke, T. Berners-Lee, M. Zereba, S. Capadisli, A. Ghanem ja A. Aboulnaga, „A Demonstration of the Solid Platform for Social Web Applications“. |
| [2] | D. Draheim, M. Felderer ja V. Pekar, „Weaving Social Software Features Into Enterprise Resource Planning Systems“. |
| [3] | „Making Web Annotations Dynamically Robust and Semantically Rich“. |
| [4] | „hypothes.is,“ [Võrgumaterjal]. Available: https://web.hypothes.is/. |
| [5] | „genius.com,“ [Võrgumaterjal]. Available: https://genius.com/web-annotator. |
| [6] | J. Pouwelse, P. Garbacki, D. Epema ja H. Sips, „The Bittorrent P2P File-Sharing System: Measurements and Analysis“. |
| [7] | B. Ford, P. Srisuresh ja D. Kegel, „Peer-­to­-Peer Communication Across Network Address Translators“. |
| [8] | B. Dodson, I. Vo, T. J. Purtell, A. Cannon ja M. S. Lam, „Musubi: Disintermediated Interactive Social Feeds for Mobile Devices“. |
| [9] | C. Holmberg, S. Hakansson ja G. Eriksson, „Web Real-Time Communication Use Cases and Requirements,“ March 2015. [Võrgumaterjal]. Available: https://tools.ietf.org/html/rfc7478. |
| [10] | M. Moore, „The semantic web: an introduction for information professionals,“ Thomson Reuters, 2011. |

Appendix

1. License

**Non-exclusive licence to reproduce thesis and make thesis public**

I, **Ioane Sharvadze**,

(*author’s name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

**Implementation of WebWeaver Platform over P2P Network**,

*(title of thesis)*

supervised by      ,

*(supervisor’s name)*

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **17.02.2018**