Distributed Discussions in Online Social Networks

Masterarbeit

Florian Müller
Betreuer: Prof. Dr. Max Mühlhäuser
Verantwortlicher Mitarbeiter: Dipl.-Inform. Kai Höver
Darmstadt, 14. August 2013

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Telekooperation
Prof. Dr. Max Mühlhüuser

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in dieser oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 14. August 2013

_____

(Florian Müller)

Zusammenfassung

Inhalt...

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

ToDo

1 Motivation

- Einleitung (Definition des Problems, Einbettung in das Forschungsfeld)
- Beschreibung der Problemstellung (Welches Problem soll erarbeitet werden?)
- Lernen in sozialen Netzwerken
- Nicht jeder ist in jeden Netzwerk vertreten
- Backup/ Synchronisierung zwischen Netzwerken

2 Grundlagen

3 Verwandte Arbeiten

- Verwendete Ansätze, Methoden und/oder Modelle (Sprachen, Entwurfsmethoden,Datenmodelle, Analysemethoden, Formalismen)

3.1 Semantic Interlinked Online Communities(SIOC)

3.2 Reclaim

Hat sich nicht jeder schon einmal vor den Rechner gesessen um, zum Beispiel, nach einen Bild gesucht das man irgendwann auf irgendeinem der unzähligen sozialen Netzwerke hochgeladen hat, einem aber partout nicht einfallen will wo? Wann und wo habe habe ich den Beitrag geschrieben, der perfekt zu meiner aktuellen Arbeit passen würde? Solche oder ähnliche Fragen wurden sicherlich schon mehrere Millionen mal von verschiedenen Menschen in der Welt des Internets gestellt. Wer hätte in so einen Fall nicht gerne alles was man über die letzten Jahre an verschiedenen Stellen im Netz geschrieben, hochgeladen oder als für ihn wichtig markiert hat zentral gespeichert um es durchsuchen zu können? Genau diesem Thema haben sich Sascha Lobo[1] und Felix Schwenzel angenommen und auf der Netzkonferenz re:publica[2] 2013 ihr gestartetes Projekt „Reclaim Social“ [2] vorgestellt.

Ziel mit diesem Projektes soziale Medien aus allen möglichen Quellen auf seinen eigenen Blog zu spiegeln und so einen zentrale Anlaufstelle für seine eigenen Inhalte schaffen. Aufbauend auf der weit verbreiteten Blogsoftware „WordPress[3]“ und der dafür vorhandenen Erweiterung „FeedWordPress“[4]. Diese Kombination ermöglichst alle Internetseiten, welche einen RSS Feed[5] anbieten, in die Datenbank von WordPress zu spiegeln. Das Problem hierbei besteht darin, dass einige sehr beliebte Internetseiten solche RSS Feeds nicht anbieten (`https://facebook.com`, `https://plus.google.com`) oder eingestellt haben (`https://twitter.com`). Für einige solcher Seiten wurden „proxy-scripte“[2, Tecg Specs Details] implementiert, welche für diese einen RSS Feed emulieren. Zugleich können in den Feeds enthaltende Medien, wie Bilder und Videos(bisher nur als Referenz), heruntergeladen und in WordPress gespeichert werden. So ist es möglich alle gespiegelten Daten einfach zu durchsuchen oder nach bestimmten Kriterien zu filtern. Zusätzlich können alle Freunde, welche auch Reclaim Social einsetzen, in einen Kontaktliste eingetragen und so auch deren Inhalte eingebunden werden.

Aktuell befindet sich dieses Projekt noch im Alpha Stadium und die Installation ist relativ kompliziert. Es ist aber geplant eine eigene Erweiterung für WordPress zu schreiben „he goal is to build just one Reclaim Social-plugin for any wordpress user“[2, How Does It Work]

1 `http://saschalobo.com/`
2 `http://re-publica.de/`
3 `http://wordpress.org/`
4 `http://feedwordpress.radgeek.com/`
5 `http://www.rssboard.org/rss-specification`

4 Anforderungsanalyse

1. Begriff soziales Netzwerk anpassen

Um sich eine Vorstellung davon zu machen, wie ein System auszusehen hat und welche Komponenten dazu nötig sind um zwei oder mehrere sozialen Netzwerke zu verbinden, soll hierzu ein kleines Ablaufbeispiel konstruiert werden.Das vollständige Ablaufdiagramm befindet sich im Anhang A.1.

4.1 Neuen Beitrag verfassen

Alles beginnt damit, dass zum Beispiel ein Student im sozialen Netzwerk A, im Forum zur Veranstaltung Telekooperation 1, eine Frage zur aktuellen Übung stellen will. Er geht zuerst in den passenden Thread und beginnt einen neuen Beitrag zu schreiben. Sobald er fertig ist, klickt er auf „Absenden“ und sein Beitrag wird in der Datenbank des sozialen Netzwerkes A gespeichert und als neuer Eintrag im Thread angezeigt (siehe Abbildung 4.1).

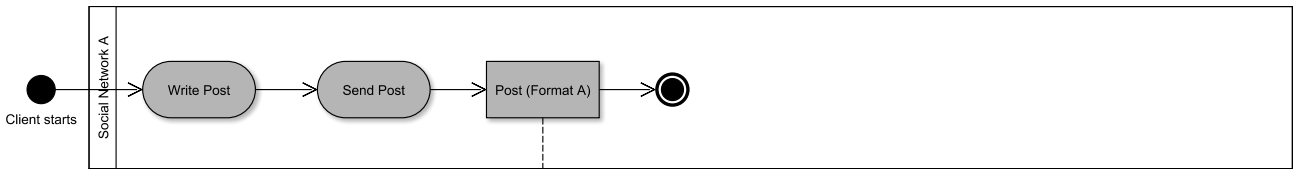

Abbildung 4.1.: Benutzer erstellt einen Beitrag im sozialen Netzwerk A.

4.2 Beiträge von sozialen Netzwerk A lesen

Um diese Beitrag in das soziale Netzwerk B transferieren zu können, müssen zuerst die Daten über eine öffentliche Schnittstelle vom Server des sozialen Netzwerks A heruntergeladen werden. Da in der Regel nicht automatisch bekannt ist, wann ein neuer Beitrag vorhanden ist, müssen die Server in zeitlichen Abständen abgefragt (polling genannt) und die zurückgelieferten Daten nach neuen Beiträgen durchsucht werden. Sind ein oder mehrere neue Beiträge gefunden worden, können diese nicht direkt an das soziale Netzwerk B geschickt werden, da sich diese in der Regel im verwendeten Datenformat unterscheiden. Diese müssen zuvor konvertiert werden.

Die einfachste Möglichkeit wäre nun die Daten von Format A nach Format B zu konvertieren. Bei zwei Formaten ist dies noch sehr einfach. Es müsste lediglich ein Konverter von Format A nach Format B und einer in die umgekehrte Richtung implementiert werden. Für den Fall, dass nun ein weiteres Netzwerk C unterstützt werden soll, würde ich die Anzahl an nötigen Konvertern auf Sechs erhöhen, wie Tabelle 4.1 zeigt.

Nimmt man an n_{sn} sei eine beliebige Anzahl sozialer Netzwerke, entspricht die Anzahl der notwendiger Konverter $n_{k1} = n_{sn} * (n_{sn} - 1)$, da für jedes Netzwerk ein Konverter in alle anderen

Tabelle 4.1.: Anzahl Konverter bei drei sozialen Netzwerken

		Nach		
		Netzwerk A	**Netzwerk B**	**Netzwerk C**
Von	**Netzwerk A**	-	×	×
	Netzwerk B	×	-	×
	Netzwerk C	×	×	-

Netzwerke erzeugt werden muss. Sollen nur Zwei oder Drei Netzwerke unterstützt werden ist der Aufwand noch sehr überschaubar, bei mehr kann dies aber sehr Aufwendig werden.

Eine elegantere Methode, welche die Anzahl zu implementierender Konverter in Grenzen halten kann, wäre die Einführung eines Zwischenformates. Geht man davon aus, dass die Daten aller Netzwerke nur in dieses Zwischenformat geschrieben und aus diesem gelesen werden müssen, würde sich der Aufwand auf maximal Zwei Konverter pro neuem Netzwerk reduzieren. Für eine beliebige Anzahl Netzwerke wären also $n_{k2} = n_{sn} * 2$ Konverter nötig. Nachteile hätte dieser Ansatz nur für $n_{sn} = 2$ und $n_{sn} = 3$, da in diesen Fällen mehr beziehungsweise gleich viele Konverter gegenüber der ersten Methode erforderlich wären. Erhöht man die Anzahl Netzwerke jedoch nur geringfügig, sinkt die Menge an Konvertern sichtbar. Für $n_{sn} = 4$ wären es $n_{k2} = 8$ statt $n_{k1} = 12$ und für $n_{sn} = 5$ ergibt sich $n_{k2} = 10$ statt $n_{k1} = 20$ Konvertern. Gleichzeitig können so syntaktische Unterschiede in den einzelnen Formaten angeglichen werden, was sie leichter handhabbar macht. Abbildung 4.2 verdeutlicht den Ablauf unter Verwendung des eben beschriebenen Zwischenformats.

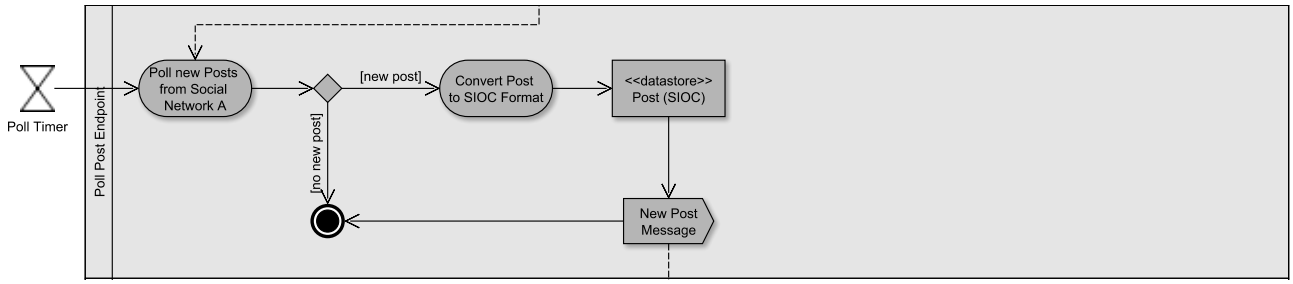

Abbildung 4.2.: Lesen des erstellten Beitrags und konvertieren in das Zwischenformat.

Liegen nun alle Daten in diesem Zwischenformat vor, könnten diese an einer Datenbank gespeichert werden. Diese Datenbank macht es möglich auf einfache Art und Weise auf die gespeicherten Daten von außerhalb zu zugreifen und diverse Abfragen ausführen zu können. Dies könnte zum Beispiel sein, in den Daten nach inhaltlich gleichen oder ähnlichen Beiträgen zu suchen oder passende externe Materialien aus Vorlesungen zu einen Thema vorzuschlagen. Die Möglichkeiten sind hier vielfältig.

Da ein Teil dieser Arbeit darin besteht Beiträge zwischen zwei oder mehr sozialen Netzwerken zu synchronisieren, muss das Eintreffen neuer Beiträge an die einzelnen Komponenten mitgeteilt werden. Da es unmöglich ist diesen Zeitpunkt vorher zu sagen ist der beste Weg einen ereignisorientierten Ansatz zu wählen. Hierbei wird beim lesen eines neuen Beitrags eine Ereignisnachricht erzeugt, welche die interessierten Komponenten über das Eintreffen informiert. So wird eine zeitliche Entkopplung von Lesen und Schreiben möglich, gleichzeitig können sich einzelnen

Komponenten an diesen Nachrichtenstrom an- oder abhängen ohne Andere zu stören. Dies erhöht die Flexibilität des Systems.

4.3 Beitrag in soziales Netzwerk B schreiben

Hat sich eine Komponenten und empfängt eine Ereignisnachricht von einen neuen Beitrag, wird ähnlich Verfahren wie schon beim Lesen, nur umgekehrt (Siehe Abbildung 4.3 links, oberer Ablauf). Der neue Beitrag wird vor den schreiben in das soziale Netzwerk B aus dem Zwischenformat in der Zielformat B konvertiert. Alle dazu nötigen Information können direkt aus der Ereignisnachricht oder zusätzlich aus der oben genannten Datenbank geholt werden.

Wünschenswert wäre es hierbei, wenn es so aussehen würde der Beitrag nicht vom hier entwickelten System, sondern von Autor des ursprünglichen Beitrags geschrieben worden. Hierzu es unerlässlich auf das Konto des Autor im entsprechenden Netzwerk zugreifen zu können, da im allgemeinen der Schreiben stellvertretend für dritte Personen nicht möglich ist. Hierzu muss zunächst für den betreffenden Autor das Konto für das soziale Netzwerk B herausgesucht werden (Siehe Abbildung 4.3 links, unterer Ablauf). Wurde ein passendes Konto gefunden, wird der konvertierte Beitrag über diesen geschrieben. Sollten kein Passender gefunden werden, müsste dies über ein vorher definiertes Konto geschehen. Dabei sollte aber auf den ursprünglichen Autor beziehungsweise Beitrag in irgendeiner Form hingewiesen werden. Dies kann zum Beispiel durch anbringen eines Links an den Text des Beitrags erfolgen.

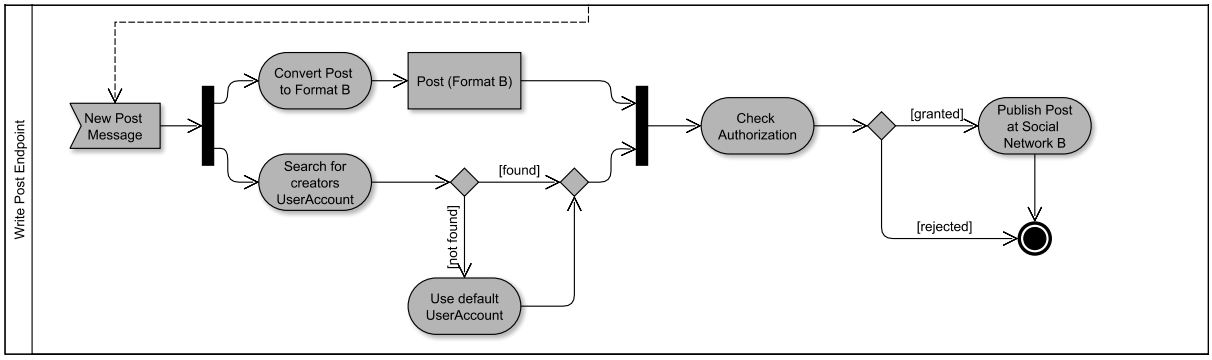

Abbildung 4.3.: Konvertierten des Beitrags in das Format B und schreiben in das soziale Netzwerk B

Gleichzeitig mit den Sammeln von Daten kommt immer auch das Thema zum Schutz der Privatsphäre auf. Wie soll damit umgegangen werden, wen ein Benutzer nicht möchte dass seine Beiträge gesammelt werden oder automatisch weiter geleiten werden sollen? Hierzu wäre es sinnvoll die eben beschriebenen Abläufe so zu erweitern, dass der Benutzer festlegen kann bestimme Quellen oder einzelne Teile für das automatische Lesen und Schreiben zu blockieren, wie es von Uldis Bojars et al. in [1] vorgeschlagen wird. Dies könnte durch eine Access Control List (ACL) realisiert werden. Hier könnte der Benutzer Lese und Schreibrechte für einzelne Bereiche festlegen, welche dann das System benutzt um einzelne Abläufe auszuführen oder abzubrechen.

4.4 Identifizierung der Komponenten

Anhand dieses kurzen Ablaufbeispiels können wir nun einige Komponenten ablesen die das System unbedingt beinhalten muss und welche ergänzend dazu Wünschenswert wären:

- Eine Komponente muss Daten von einen sozialen Netzwerk in das System einlesen und diese in ein geeignetes Zwischenformat konvertieren können.
- Eine weitere Komponente nimmt Beiträte im Zwischenformat entgegen, konvertiert diese in das Format des entsprechenden Netzwerkes und schreibt diese dorthin.
- Um stellvertretend für einen Benutzer schreiben zu können muss es möglich sein nach Konto eines Benutzer in einem Netzwerk suchen zu können.
- Um die Privatsphäre der Benutzer zu wahren, wäre eine ACL Mechanismus sinnvoll.

5 Design

- Die sollen die Anforderungen aus Kapitel 4 letztendlich in ein funktionstüchtiges Programm umgesetzt werden.

5.1 Wahl des Zwischenformats

- Warum kein eigenes Format
- Warum nicht RSS
- Warum nicht Dublincore + X
- Warum SIOC + FOAF + X

5.1.1 RSS

5.1.2 Dublin Core

5.1.3 SIOC und FOAF

5.2 Social Online Community Connectors

Einleitung über den Sinn und Zweck eines Connectors in SOCC

5.2.1 Aufbau

5.2.2 SIOC Service Auth Ontology

5.2.3 SOCC ConnectorCFG Ontology

5.2.4 W3 Web ACL Ontology

5.3 Routen der Einträge

- Eigener Ansatz mit JMS
- Apache Camel

6 Implementierung

- Probleme und deren Lösung während der Umsetzung
- Beispiel: Zugriff auf Moodle über Webservice

6.1 Verwalten SIOC Daten

- Sesame, Jena -> RDF2Go
- In Memory, XML/RDF File, Tripplestore

6.2 Zugriff auf Online Netzwerke und Abbildung in SIOC

6.2.1 Moodle

6.2.2 Canvas LMS

6.2.3 Facebook

6.2.4 Google Plus

6.2.5 Youtupe

6.3 Implementieren der Connectoren

-

7 Fazit

Literaturverzeichnis

[1] U. Bojars, J. Breslin, and S. Decker. Porting social media contributions with SIOC. *Recent Trends and Developments in Social Software*, 6045:116–122, 2011.

[2] F. Schwenzel and S. Lobo. Reclaim social. http://reclaim.fm/, 2013. Letzter Zugriff: 14. August 2014.

A Anhang

A.1 Anforderungsanalyse Ablaufdiagramm