Recherchebericht

18. April 2011

Inhaltsverzeichnis

1 Glossar **2**
- 1.1 Anticloud . . . 2
- 1.2 Distributed Hash Table . . . 2
- 1.3 Föderation . . . 2
- 1.4 Gruppe . . . 2
- 1.5 Kollaborative Elemente . . . 2
- 1.6 Kooperative Komunikationsräume . . . 2
- 1.7 Notarfunktionalität . . . 3
- 1.8 Peer-to-Peer . . . 3
- 1.9 Technosoziales System . . . 3
- 1.10 XMPP . . . 3

2 Konzepte **3**
- 2.1 Einführung . . . 3
- 2.2 Kollaboration . . . 3
- 2.3 Architekturen . . . 4
- 2.4 Föderation . . . 4
- 2.5 XMPP . . . 4
- 2.6 Rechtekonzept . . . 5
- 2.7 Datenreplikation . . . 5

3 Komparative Analyse verteilter sozialer Netzwerke **5**
- 3.1 Friendika . . . 5
  - 3.1.1 Zugriffskontrolle . . . 6
  - 3.1.2 Datenreplikation . . . 6
- 3.2 Apache Wave . . . 7
  - 3.2.1 Verteilte Datenstruktur . . . 7
  - 3.2.2 Zugriffskontrolle . . . 7
  - 3.2.3 Datenreplikation . . . 8

3.3 Diaspora 8
3.3.1 Zugriffskontrolle und Datenreplikation 8
3.4 Buddycloud 8
3.4.1 Datenreplikation 9
3.4.2 Zugriffskontrolle 9

1 Glossar

1.1 Anticloud

Eine cloudähnliche Anwendung, die so realisiert ist, dass Daten nicht zentral gesammelt werden, sondern die Nutzer möglichst viel Kontrolle über diese behalten und es keinen zentralen Server gibt. Im Gegensatz zur klassischen Cloudarchitektur ist zu jedem Zeitpunkt klar auf welchen Knoten im Netz sich die Daten eines Nutzers befinden bzw. wohin sie repliziert werden dürfen.

1.2 Distributed Hash Table

Verteilte Hashtabellen sind eine weitverbreitete Datenstruktur zur Lokalisierung von Daten in →*Peer-to-Peer*-Netzwerken. Sie stellen Dezentralisierung in den Vordergrund und stellen einen niedrigschwelligen Ansatz der Datenhaltung dar.

1.3 Föderation

Das Zusammenschließen einzelnder Serverinstanzen, um Daten und Dienste gemeinsam nutzen zu können. Im Gegensatz zu →*Peer-to-Peer*-Netzwerken kommunizieren Nutzer nicht direkt miteinander, sondern nutzen föderierte Server. Das wahrscheinlich bekannteste Beispiel einer solchen Infrastruktur ist Email.

1.4 Gruppe

Ein Zusammenschluss von Akteuren, in unserem Kontext Firmen, um bezüglich eines Projekts einen →*kooperativen Kommunikationsraum* zu bilden.

1.5 Kollaborative Elemente

Technisch realisierte Funktionen, die die Zusammenarbeit unterstützen. Beispiele hierfür sind Abstimmungs- und Versionskontrollsysteme.

1.6 Kooperative Komunikationsräume

Zusammenhänge, in denen Akteure zum gemeinsamen Profit kommunizieren.

1.7 Notarfunktionalität

Die Möglichkeit, Artefakte, ggf. unter der Aufsicht Dritter technisch zu bestätigen bzw. zu unterschreiben, sodass diese Bestätigung insbesondere von anderen zur Kenntnis genommen werden kann und manipulationssicher ist.

1.8 Peer-to-Peer

Eine flache Netzarchitektur, die aus mehreren gleichberechtigten Knoten besteht, welche ohne eine zentrale Instanz miteinander kommunizieren können. Bei verteilten sozialen Netzwerken impliziert dies, dass jeder Nutzer seine persönlichen Daten auf einem eigenen Peer (im Normalfall Consumer-Hardware) vorhält und demensprechend volle Kontrolle über Replikation und Zugriff besitzt.

1.9 Technosoziales System

Ein System, das sowohl aus technischen als auch aus sozialen Elementen zusammengesetzt ist.

1.10 XMPP

Das *Extensible Messaging and Presence Protocol* ist ein offenes Instant Messaging Netzwerkprotokoll, das aber aufgrund seiner Erweiterbarkeit mittlerweile auch für VoIP und andere Anwendungen genutzt wird. Mehrere verteilte soziale Netzwerke nutzen XMPP als Grundlage, da es bereits Authentifizierung, Verschlüsselung und eine →*föderierte* Serverarchitektur bereitstellt.

2 Konzepte

2.1 Einführung

Mit dem drastischen Wachstum und der zunehmenden Monopolisierung sozialer Netzwerke in den letzten Jahren rücken Privatsphäre und die kommerzielle Verwendung der gesammelten Nutzerprofile immer mehr in den Vordergrund. Zusätzlich stellen der rasante Anstieg der Nutzerzahlen Anbieter vor Probleme, da zentralisierte Lösungen inhärente Einschränkungen in der Skalierbarkeit aufweisen.

Verteilte soziale Netzwerke versuchen eine Lösung all dieser Probleme zu sein. Ein essentieller Aspekt dieser Neuentwicklungen ist die Veröffentlichung des Quellcodes unter einer Freien Software Lizenz wie der GPL.

2.2 Kollaboration

Soziale Netzwerke lassen sich grundsätzlich in zwei Aspekte aufteilen, Kommunikation und Kollaboration. Während die kommunikative Funktionalität bei den meisten Netzwerken voll ausgearbeitet ist, sind kollaborative Funktionen oft noch in den Kinderschuhen.

Einige weitverbreitete kollaborative Anwendungen sind Wikis, Versionskontrollsysteme wie *git* oder *subversion*, Abstimmungsmechanismen wie *doodle* und Bugtracker wie *Bugzilla*. Zusätzlich existiert Software wie *trac*, welche mehrere dieser Aspekte vereint und nahtlose Integration von Komponenten wie Versionsverwaltung und Bugerfassung ermöglicht.

Kollaborative Anwendungen stellen für verteilte soziale Netze eine zusätzliche Herausforderung dar, da sie, von Versionskontrollsystemen abgesehen, fast ausschließlich für Client-Server-Betrieb ausgelegt sind.

2.3 Architekturen

Während es eine Vielzahl von Ansätzen zur Verteilung der Nutzerdaten und Zugriffskontrolle gibt, kristallisieren sich mittlerweile zwei Architekturen heraus.

Peer-to-Peer-Netzwerke bestehen aus einer Menge von egalitären Knoten, die zumeist auf lokaler Customerhardware laufen. Zudem ist eine Instanz des sozialen Netzes meist mit einem einzelnen Nutzer assoziiert. Bestehende P2P-Systeme zeichnen sich jedoch durch eine mangelnde Maturität aus und können für einzelne Anwender einen nicht vertretbaren Wartungsaufwand darstellen.

Föderation ist eine erprobte Technologie, die durch teilweise Zentralisierung die Vorteile von Peer-to-Peer-Netzwerke und klassischen Server-Client-Architekturen verbindet.

2.4 Föderation

In einem föderierten Netzwerk kommunizieren Anwender mit Servern, welche wiederum miteinander kommunizieren. Dies erlaubt eine teilweise Zentralisierung von Dienstleistungen mit den assoziierten Vorteilen, gibt Anwendern aber gleichzeitig die Möglichkeit, durch eigene Infrastruktur dem Netzwerk beizutreten.

Das beste Beispiel eines solchen Netzes von Server ist Email über SMTP. Anwender können aus einer Menge von Dienstleistern wählen oder aber eigene Server betreiben und uneingeschränkt am Gesamtnetz teilnehmen. Je nach Anforderungen gibt es Anbieter, die verschiedene Leistungen unentgeltlich oder kommerziell anbieten.

Dieses Modell ist eine Grundlage das Netz sowohl für professionelle Anwender (die wahrscheinlich einen höheren Wert auf Privatsphäre legen) als auch für technisch unversierte Nutzer (deren Preferenzen bei der unmittelbaren Benutzbarkeit liegen) attraktiv zu gestalten.

2.5 XMPP

Mehrere Projekte nutzen XMPP als Grundlage, da bereits eine ausgebaute Serverinfrastruktur und verschiedene ausgereifte Implementationen existieren. Das Protokoll unterstützt Erweiterungen wie zusätzliche Services, wodurch das Entwickeln eines funktionsfähigen sozialen Netzwerks auf das Ergänzen fehlender Funktionalität beschränkt wird.

Kritische Aspekte wie Föderation und Datenhaltung wurden hier bereits zufriedenstellend gelöst.

Durch die Modularisierung des Protokolls kann eine nahezu beliebige Verteilung von Diensten ermöglicht werden. Zum Beispiel muss nicht jeder Serverbetreiber einen eigenen Verzeichnisdienst an seinen XMPP Server anbinden; Anwender können sich aus den angebotenen Diensten verschiedener Server ein für sie optimales Profil zusammenstellen.

2.6 Rechtekonzept

Eine verständliche, jedoch feingranulare Zugriffskontrolle ist ein essentieller Bestandteil eines jeden Systems, das private Daten verarbeitet. Anwender können im Normalfall mit einer Fülle von Optionen entscheiden, ob erzeugte Daten grundsätzlich global oder auch nur für sie persönlich sichtbar sein sollen. Grundsätzlich darf nur der *Besitzer* der Daten Einfluss auf Zugriffe haben und sollte durch sinnvolle sinnvolle Voreinstellungen in dieser Aufgabe unterstützt werden.

Wenn kollaborative Elemente inkorporiert werden, muss zwangsläufig auch eine Verwaltung der Schreibrechte geschehen. Dies kann analog zu Leserechten durch den „Besitzer“ einer Ressource erfolgen oder durch eine Gruppe von Administratoren oder Moderatoren. Viele Systeme erlauben eine gestaffelte Vergabe von Rechten an Moderatoren, sodass nicht unbegrenzt Vertrauen in Dritte gesetzt werden muss. So kann zum Beispiel eingeschränkt werden, dass Moderatoren wiederum Moderatorenrechte an Dritte vergeben oder Moderatoren können nur Lese- aber keine Schreibrechte vergeben.

2.7 Datenreplikation

In verteilten sozialen Netzwerken sollte die Datenreplikation dem Muster der Anticloud folgen und muss dementsprechend tief in dem Rechtekonzept verankert sein. Der Anwender muss zwingend zu jedem Zeitpunkt volle Kontrolle über die Verbreitung seiner Daten in der Wolke besitzen.

Verbunden mit den Vorteilen der föderierten Serverarchitektur, einer zuverlässigen Authentifizierung, eines mächtigen Autorisierungskonzepts und voller Kontrolle über die Verbreitung persönlicher Daten hat der Anwender eine bisher unübertroffene Kontrolle über persönliche Daten in sozialen Netzwerken ohne Vertrauen in Dritte setzen zu müssen.

3 Komparative Analyse verteilter sozialer Netzwerke

3.1 Friendika

Friendika ist ein verteiltes soziales Netzwerk mit den dazugehörenden klassischen Funktionen. Man kann mehrere Profilseiten haben, bei denen die Sichtbarkeit auf bestimmte User beschränkt werden kann. Es gibt private und öffentliche Gruppen. Auf andere soziale Netzwerke kann via Plugins zugegriffen werden. Weitere Features sind Micro-blogging, Photo/Youtube video sharing, Server-to-server message encryption, Authentifikation mit OpenID, Like/Dislike. Upgrades werden simpel durch herunterladen der neuen Dateien realisiert. Durch Plugins kann die Funktionalität erweitert werden wie z.B. calc - ein

Taschenrechner. Quelltextkommentare sind eher spärlich. Posts koennen auch gelöscht werden und sollen dabei auch bei den anderen Friendika Instanzen gelöscht werden. Zur Zeit gibt es keine Funktionen zur geschäftlichen Kooperation.

Der zentrale Datenspeicher jeder Friendika Instanz ist ihre lokale Datenbank, die als MySQL-Datenbank geplant ist. In diesen Datenbanken sind alle Daten der lokalen Benutzer einer Friendika Instanz enthalten, wie z.B. Status, Bilder, Nachrichten. Es werden ausserdem alle Daten, auf die ein lokaler Benutzer bei einer anderen Friendika Instanz Zugriff hat, in der lokalen Datenbank gespeichert. So entsteht eine Redundanz eines Teils aller Daten des gesamten Friendika Netzwerkes. Andererseits sind alle Daten, auf die nur lokale Benutzer Zugriff haben, auch nur in der lokalen Datenbank verfuegbar.

3.1.1 Zugriffskontrolle

Um in Friendika den Zugriff auf gewisse Daten zu regeln, gibt es drei grundlegende Elemente, Kontakte, Profile und Gruppen. Ein Benutzer kann einen anderen Benutzer bitten ihn als Kontakt hinzuzufügen. Der angefragte Benutzer kann nun entscheiden, ob er den fragenden Benutzer als Fan oder als Freund hinzufügen will. Als Fan sieht der fragende Benutzer die Nachrichten dieses Kontaktes und dieser Kontakt kann ihm private Nachrichten schicken, jedoch nicht umgekehrt. Als Freund können beide Benutzer miteinander kommunizieren. Ausserdem besteht die Möglichkeit Freunde zu ignorieren bzw. zu blockieren. Beim Ignorieren erhält man keine Nachrichten mehr von diesem Freund und derselbige weiß dies nicht. Im Gegensatz dazu ist ihm das beim Blockieren bewusst.

Ein Benutzer kann beliebig viele Profile haben, dabei hat er immer ein globales Profil, welches von jedem eingesehen werden kann. Die Sichtbarkeit jedes weiteren Profils bestimmt der Besitzer, indem er seinen Kontakten Zugriff darauf gewährt. Für jedes Profil kann entschieden werden, ob es im lokalen Verzeichnis und/oder im globalen Verzeichnis sichtbar ist. Das lokale Verzeichnis enthält Profile der lokalen Friendika Instanz, während das globale Verzeichnis Profile des gesamten Friendika Netzwerkes enthält. Gruppen bestehen aus Freunden. Ein Mitglied einer Gruppe kann für jede seiner Nachrichten bestimmen, ob sie für die Gruppe sichtbar sind.

3.1.2 Datenreplikation

Bei der Datenreplikation können zwei unterschiedliche Konzepte eingesetzt werden, Polling und das Prinzip publish/subscribe. Beim Polling lädt eine Friendika Instanz in regelmäßigen Abständen die neuen Nachrichten bei den anderen Instanzen herunter. Publish/subscribe bedeutet, dass sich eine Friendika Instanz bei einer anderen anmeldet, um informiert zu werden, wenn neue Nachrichten vorhanden sind. Falls eine Instanz auf diese Art informiert wird, lädt sie die neuen Nachrichten herunter. Da so die Latenz zwischen Veröffentlichung einer Nachricht und Ausbreitung derselben im Gegensatz zum Polling stark verringert wird, wird publish/subscribe bevorzugt benutzt.

Nachrichten lassen sich aufteilen in private und öffentliche. Beim Senden einer privaten Nachricht werden der/die Empfänger gemäß publish/subscribe benachrichtigt. Falls

beim Senden eine Zeitintervallsüberschreitung auftritt, also die Nachricht wahrscheinlich nicht ankommt, wird sie in eine Warteschlange zum erneuten Senden eingereiht. Die Übertragung erfolgt mittels des DFRN Protokolls und ist verschlüsselt. Öffentliche Nachrichten werden nicht verschlüsselt, da sie jeder lesen können soll. Um öffentliche Nachrichten zu übertragen, wird PuSH (pubsubhubbub) verwendet, welches auch dem Prinzip von publish/subscribe folgt. Falls PuSH nicht funktionierten sollte, wird auf Polling via DFRN zurückgegriffen.

3.2 Apache Wave

Wave bezeichnet sowohl die Produkte Google/Apache Wave als auch einen Zusammenhang, über den Kontakte innerhalb der Software miteinander interagieren. Wave bietet Kommunikation und Kollaboration in Echtzeit an. Benutzer kommunizieren innerhalb einer Wave miteinander und können in dieser wie in einer Mischung aus Chat, Email, Forenbeitrag und Wikiartikel agieren, wobei die Übergänge fließend sind. Beispielsweise könnte eine Wave als Chat starten, wobei im Verlauf mehrere Teilnehmer anfangen einen gemeinsamen Artikel zu schreiben, sodass die Chatfunktionalität in den Hintergrund tritt und am Ende ein fertiger Artikel kollaborativ entstanden ist. Eine Wave kann neben formatiertem Text auch Bilder, Musik, Videos, Orte in Googlemaps, Abstimmungen, etc. enthalten. Die multimedialen Inhalte einer Wave sind über sog. Gadgets, die in einem einfachen XML-Format geschrieben sind, quasi beliebig erweiterbar (In der Tat sind u.a. ganze Spiele denkbar). Wave benutzt XMPP für Föderation. Das Projekt wurde ursprünglich von Google angestoßen und wird seit 2009 von der Apache Foundation weiterentwickelt.

3.2.1 Verteilte Datenstruktur

Wave Server greifen auf MongoDB als Datenbank zurück oder behalten alle Daten nur flüchtig vor. Kontaktdaten werden zentral gespeichert, Waves dezentral. Das heißt, dass ein bestimmter Kontakt immer an seinen Server gebunden ist, während eine Wave auf allen Servern abgelegt wird, die beteiligte Kontakte an dieser Wave haben.

3.2.2 Zugriffskontrolle

Jeder Kontakt ist in der Lage auf seinem Server eine Wave zu starten, er gehört dann automatisch zu dieser Wave. Jeder Kontakt, der zu einer Wave gehört, kann selbstständig weitere Kontakte hinzufügen oder bestehende entfernen. Wird ein Kontakt zu einer Wave hinzugefügt, erhält er den gesamten Verlauf aller Änderungen der Wave. Wird ein Kontakt von der Wave entfernt, bekommt er keine weiteren Updates mehr zu dieser. Diese Zugriffskontrolle hat also keine Rechtedelegation oder unterschiedliche Rollen, vielmehr wird nur unterschieden ob Kontakte an einer Wave mitarbeiten oder nicht.

3.2.3 Datenreplikation

Nehmen mehrere Kontakte von unterschiedlichen Servern an einer Wave teil, so schickt jeder Kontakt seine Änderungen zuerst an seinen Server, der diese dann auch an die Server der anderen Kontakte weiterleitet. Dadurch ist gewährleistet, dass jeder erreichbare Server stets den aktuellen Zustand der Wave lokal vorrätig hat. Technisch greift dieses Vorgehen auf XMPP zurück.

3.3 Diaspora

Diaspora ist ein dezentrales Open-Source soziales Netzwerk basierend auf Ruby on Rails und MongoDB. Es wurde 2010 von vier New Yorker Studenten unter dem Hintergrund der Datenschutz-Probleme von Facebook initiiert und als Prealpha am 15.9.2010 veröffentlicht. Fortan wird Diaspora – immernoch Alphaphase – fortschreitend durch die Herausgeber als auch von freiwilligen Helfern weiterentwickelt. Es bestehen seit Februar 2011 bereits Schnittstellen zu Facebook, über die man Daten (wie Fotos) übermitteln kann.

3.3.1 Zugriffskontrolle und Datenreplikation

Besondere Grundsätze die die Privatsphäre erhöhen sollen sind vor allem durch die dezentrale Verteilung der Daten und die Einteilung der Kontakte in äspects"gegeben. So sollen die eigenen Daten verschlüsselt auf dem eigenen Pod und den verbundenen Diaspora*-Kontakten liegen, statt auf einem zentralen Server wie bei anderen Anbietern.

Aspekte dienen dazu jedem Kontakt eines *Seeds* (Accounts) ein bestimmtes „Gesicht“ zu zeigen. So wäre es denkbar Aspekte für verschiedene Tätigkeiten und Privates zu haben und diese Facetten im sozialen Netz quasi zu trennen.

Es existiert zur Zeit keine Spezifikation der Systemarchitektur, Zugriffskontrolle oder des Inter-Pod-Protokolls, wodurch eine Evaluierung dieser Eigenschaften ausgesprochen schwierig ist. Erste veröffentlichte Versionen des Quellcodes wiesen starke Mängel sowohl in Konzeption wie auch Qualität auf, sodass davon ausgegangen werden muss, dass ein irgendwie geartetes Sicherheitskonzept nichtexistent ist bzw. mangelhaft implementiert wurde.

Da der Quellcode zusätzlich unter rasanter Entwicklung steht lassen sich keine verlässlichen Aussagen über Datenreplikation im Netz treffen. In aktuellen Versionen werden zu versendende Nachrichten mit einem Account-spezifischen RSA-Schlüssel verschlüsselt und auf fremden Pods mittels eines Publish-Subscribe-Verfahrens repliziert.

3.4 Buddycloud

Buddycloud ist ein auf Jabber/XMPP basierendes verteiltes soziales Netzwerk. Die grundlegende soziale Komponente des Netzes sind sogenannte Channels, vergleichbar mit individuellen Facebook Walls, welche in privat und themenbasiert unterteilt werden. Private Channels sind fest an eine spezifische Jabber-ID gebunden und erlauben

zusätzliche Features wie Geolokation. Buddycloud enthält zur Zeit keine Funktionen für kollaboratives Arbeiten.

Auf der technischen Ebene basiert Buddycloud auf dem in XEP-0060 spezifizierten publish-subcribe Mechanismus und erweitert ihn um Komponenten, die soziale Netze unterstützen. pub-sub erlaubt es XMPP-Entitäten Nodes auf einem pub-sub-Service zu erzeugen und Daten an diese zu senden. Alle Subscriber werden dann automatisch von neuem Content in Kenntnis gesetzt. Ein sogenannter Channelserver verwaltet alle lokalen Channel und implementiert Aspekte wie Zugriffskontrolle etc. In naher Zukunft soll die Spezifikation des Channelprotokolls als eigenständiges XEP (XMPP Extension Protocol) veröffentlicht werden. Channelinhalte werden nach dem *ATOM* formatiert versendet und abgespeichert.

Es existieren zur Zeit drei Implementierungen des Channelprotokolls, ein Node.js basierter Server der an ejabberd angebunden werden kann, ein Modul für Prosody und ein veralteter stand-alone Server. Es sind Clients für Web, Android und iPhone in Planung beziehungsweise im frühen Entwicklungsstadium.

Die Node.js Implementation des Channelservers speichert sowohl den eigentlichen Channelinhalt als auch die channelspezifischen Metadaten, die entweder in der dokumentenorientierten Datenbank *CouchDB* oder in einer objektrelationalen *PostgreSQL* Datenbank abgelegt werden.

Zusätzliche Services können über andere XMPP Komponenten bereitgestellt werden. So wird zum Beispiel Geolokation mittels eines sogenannten Location-Butlers nach XEP-0255 bereitgestellt. Kollaborative Elemente würden in Zukunft wahrscheinlich als von Buddycloud grundsätzlich unabhängige Komponenten in XMPP implementiert werden.

3.4.1 Datenreplikation

Bei der Datenreplikation wird strikt nach dem Publish-Subscribe-Modell vorgegangen. Ein Nutzer meldet sich beim Channelserver an den für ihn interessanten Channels an (Subscribe) und wird bei neuen Inhalten (Publish) automatisch über diese informiert. Diese Nachricht kann einen kurzen Abriss oder sogar die kompletten neuen Channelinhalte enthalten. Dieses Verfahren minimiert die Latenz zwischen dem Publizieren von Inhalten und dem Empfang beim Rezipienten.

Die Kommunikation zwischen Server und Client läuft falls konfiguriert per TLS verschlüsselt ab. Im Normalfall ist dann auch die Kommunikation zwischen den Servern per TLS abgesichert. Dies hat alle Nachteile einer Hop-to-Hop-Verschlüsselung und setzt ein Vertrauen in die Server- und Clientkomponente vorraus. Es gibt jedoch mehrere Verfahren (OpenPGP, OTR) XMPP-Kommunikation End-to-End abzusichern. Inwieweit End-to-End-Verschlüsselung mit dem Grundgedanken sozialer Netzwerke kompatibel oder überhaupt erwünscht ist, ist zu evaluieren.

3.4.2 Zugriffskontrolle

Buddycloud Channels lassen sich nach zwei Zugriffsmodellen verwalten: Whitelist und Offen. Im Whitelist-Modell können nicht-authorisierte Dritte nur den Channeltitel und

Beschreibung lesen. Jabber-IDs können von Channelmoderatoren zur Whitelist hinzugefügt werden und danach Channelinhalte, Geolokationsdaten etc. abfragen. Im offenen Modus kann jeder allen Inhalten des Channels folgen.

Nachdem eine Jabber-ID einem Channel folgt, sind Moderatoren in der Lage verschiedene diesem Nutzer verschiedene Rechte einzuräumen. Es gibt Lese-, Schreib- und Löschrechte. Nutzer können auch komplett aus einem Channel verbannt werden. Zukünftige Rechteanfragen werden dann automatisiert abgelehnt.