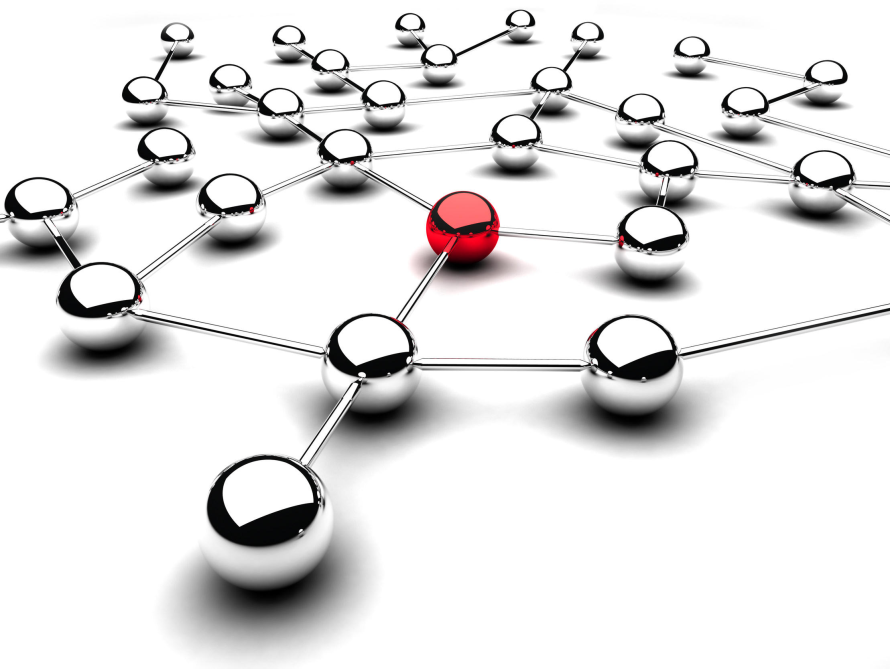
IRC Services in C

Diensteserver für einen Internet-Chat



Zürcher Hochschule für angewandte Wissenschaften

Betreuungsperson: Karl Brodowsky, Fachdozent für Systemprogrammierung

Erscheinungsjahr: 2015

Eine Semesterarbeit von Severin A. Müller

Danksagung

Ich möchte mich bei allen bedanken, die mich während der Erstellung dieser Arbeit in irgendeiner Weise unterstützt haben. Namentlich sind dies:

Lis Steiger, lic. phil. für das Korrekturlesen der Arbeit

Sandro Preisig für den Input von Anforderungen

Bettina Hegenauer für die Unterstützung während der Arbeit

Inhalt

1. Einleitung
2. Einführung in irc und services
3. marktanalyse
4. anforderungen und architektur
   1. anforderungen
   2. use case
   3. architektur
5. umsetzung dienste im einzelnen
   1. basis server
   2. nickserv
   3. chanserv
   4. botserv
   5. operserv
   6. adminserv
6. testing
7. ausblick
8. fazit
9. anhang

# 1 Einleitung

## 1.1 Themenwahl

Wir leben im Zeitalter von Web 2.0 wodurch der klassische Text-Chat an Bedeutung verloren hat. Schliesslich sind Web-Browser heute fähig Videos abzuspielen und Webcam Übertragungen durchzuführen. IRC als reiner Text-Chat spielt also nur noch eine untergeordnete Rolle bei den Anwendern.

Dennoch ist der IRC (Internet Relay Chat) in UNIX-Kreisen immer noch äusserst populär. Nehmen wir zum Beispiel den Chat auf irc.freenode.org. Dort finden sich auch heute noch zehntausende Benutzer die gleichzeitig online sind. Sie treffen sich themenbasierten Räumen wie zum Beispiel #linux oder #C und bilden so eine riesige Gemeinschaft um bei Fragen zu den jeweiligen Themen zu helfen.



Abb. 1: Chatraum auf irc.freenode.org

Auch wenn die Benutzer heute auf andere Chat-Technologien ausweichen heisst das nicht, dass der IRC am Aussterben ist. Viele Benutzer in der Linux-Welt machen sich nichts aus grafisch hochstehenden Flash-Chats sondern möchten schnell und einfach Informationen austauschen.

Da jedoch nicht immer alle Benutzer freundliche Absichten haben und sich teilweise daneben benehmen benötigt man Mittel um sich solch ungebetener Gäste zu entledigen. Dazu dienen sogenannte Operatoren in Räumen. Sie haben Rechte, die „normale“ Benutzer nicht haben. So können Operatoren zum Beispiel einen Benutzer „kicken“, also aus dem Raum werfen oder „bannen“ also verbannen, sodass dieser Benutzer den Raum nicht mehr betreten kann.

Das Protokoll auf dem IRC zugrunde liegt sieht vor, dass der erste Benutzer der den Raum betritt automatisch den Operatoren-Status erhält. Er kann weitere Operatoren bestimmen und hat weitere Rechte (näheres dazu im Kapitel ).

Dadurch entsteht jedoch das Problem, dass wenn nur ein Operator im Raum ist und er diesen verlässt niemand mehr Operatoren-Rechte hat, bis alle weiteren Benutzer den Raum verlassen und dieser so neu erstellt werden kann.

Ein weiteres Problem ist, dass das Protokoll keine Speicherung von Benutzernamen (sog. „Nickname“) zulässt. Wenn also ein Benutzer den gewünschten Namen bereits innehat gibt es keine Möglichkeit denselben Namen zu verwenden.

Um diesen und weiteren Problemen zu begegnen wurden sogenannte „Services“ erschaffen. Diese bieten zum Beispiel die Möglichkeit, Nicknames oder Chaträume (sog. Channels) zu registrieren und so vor unbefugtem Zugriff zu schützen.

In einigen bestehenden Lösungen sind diese bereits enthalten, jedoch gibt es zahlreiche Server, die ohne Services ausgeliefert werden.

Der Autor der vorliegenden Arbeit war lange Zeit als Moderator auf dem Chat von Swisscom (damals Bluewin) tätig. Dieser Chat lief auf einer proprietären Lösung mit integrierten Services. Nach der Abschaltung des Chat begannen einige Nutzer eigene Chats aufzusetzen, dies mit Open-Source Lösungen.

Da viele Open-Source Lösungen keine integrierten Services haben, stellte sich so bald die Frage nach einer entsprechenden Lösung. Es gibt einige gute und populäre Services-Lösungen, jedoch unterscheiden diese sich stark und viele Benutzer wünschten sich, dass die Befehle ähnlich wie auf dem Bluewin Chat aufgebaut seien. Um dieses Bedürfnis zu befriedigen entschloss sich der Autor eine entsprechende Lösung zu entwickeln. Die vorliegende Arbeit befasst sich mit dieser Lösung.

## 1.2 Ziele der Arbeit

Um die im Kapitel 1.1 beschrieben Probleme zu lösen und die Bedürfnisse zu befrieden soll ein Services-Server geschaffen werden, der sich zu einem bestehenden IRC Chat verbindet und diese Dienste anbietet. Die Umsetzung soll in der Programmiersprache C erfolgen. Namentlich soll folgende Dienste angeboten werden:

* **Nickserv**: Dienst für die Registrierung eines Nickname um diesen mittels Passwort vor der Verwendung durch andere zu schützen.
* **Chanserv**: Dienst für die Registrierung eines Channels. Zudem wird die Möglichkeit geschaffen, Operatoren zu registrieren, die sich in einem registrierten Channel von Chanserv die Operatoren Berechtigung selber geben können.
* **Botserv**: Dienst für die Erstellung von Bots die einen Channel offenhalten sollen.
* **Operserv**: Dienst für Benutzer mit erweiterten Rechten (IRC Operatoren, Administratoren).
* **Adminserv**: Dienst für Administratoren um bestimmte Operationen wie z.B. das Speichern der Datenbank vor einem Neustart durchzuführen.

Um keine Abhängigkeit von einem Datenbankserver zu erzeugen sollen die Daten in einer Sqlite-Datenbank gespeichert werden. Sqlite eine Datenbank die Dateibasiert ist und SQL Syntax unterstützt. Der gesamte Datenbestand soll beim Starten des Servers in den Arbeitsspeicher geladen und in regelmässigen Abständen gespeichert werden.

Die spezifischen Einstellungen des Servers und der Dienste sollen konfigurierbar sein. Des Weiteren soll für jeden Befehl der gesendet werden kann eine Hilfedatei erstellt werden die den jeweiligen Befehl erklärt.

## 1.3 Aufgabenstellung

Um die in Kapitel 1.2 genannten Ziele zu erreichen werden folgende Teilschritte durchgeführt:

* Marktanalyse: Feststellung bereits existierender Lösungen
* Studium des IRC Protokolls (RFC 1459) um die protokollarischen Anforderungen zu erfassen.
* Aufnehmen der Anforderungen: Welche Befehle soll ein Dienst unterstützen und wie sollen diese funktionieren?
* Festlegung der Grenzen: wie viele Einträge sollen Listen maximal haben dürfen?
* Erarbeiten der Grundlagen für Timer und Signale welche für Timeouts benötigt werden.
* Durchführung des Requirement Engineering
* Ausfertigung des technischen Entwurfs
* Implementierung Basis-Server
* Implementierung der Dienste
* Testing inkl. Ausarbeitung eines Testkonzepts und Testprotokolls
* Dokumentation

# 2 Einführung in IRC und Services

## 2.1 Geschichte des IRC

Das IRC (Internet Relay Chat) stammt noch aus den Anfängen des heutigen Internets. Um 1988 wurde das ARPAnet außer Betrieb genommen und das NSFnet übernahm die Funktionen. Hierdurch wurde das TCP/IP Protokoll erschaffen und auseinander liegende Institutionen konnten miteinander kommunizieren.1

Die ursprüngliche Idee eines Chat-Netzwerkes entstand im BITNET unter dem Namen Relay Chat. Dieses System wurde vom finnischen Studenten Jarkko Oikarinen, der an der Fakultät für Informatik der Universität Oulu studierte, im Sommer 1988 auf das Internet übertragen.2

Durch einen Freund Oikarinens, erfuhren Universitäten in den Vereinigten Staaten von dem Chat. Es folgte eine Anfrage, ihren Server mit dem finnischen zu verbinden. So verliess der IRC zum ersten Mal Finnland.

Bereits 1989 gingen die ersten deutschen Universitäten mit Chats ans Netz und einige geschichtliche Ereignisse trugen in der Vergangenheit in besonderem Maße zur Popularität des IRC bei. Während des Golfkrieges im Jahr 1991 waren die aktuellsten Berichte über den Verlauf des Krieges über den IRC erhältlich. Ein ähnliches Szenarium spielte sich im September 1993 ab, als gegen das sowjetische Staatsoberhaupt, Boris Yeltsin, geputscht wurde. IRC-Benutzer aus Moskau erzählten live von den dortigen Ereignissen.3

Im Jahr 2000 ging in der Schweiz der Bluewin Chat online. Zu ihren besten Zeiten waren über 6‘000 Benutzer gleichzeitig online und bildeten so den populärsten Chat der damaligen Zeit.

Seit Mitte der 2000er Jahre die Flash basierten Chats an Popularität gewannen gingen die Nutzerzahlen auf dem IRC Netzwerken zurück. Die grossen Netzwerke wie zum Beispiel QuakeNet oder FreeNode erfreuen sich jedoch immer noch grosser Beliebtheit.

## 2.2 Funktionsprinzip des IRC

Mehrere IRC Server können sich zu einem Netzwerk zusammenschliessen. So können die Benutzer die Benutzer von allen Servern miteinander agieren.

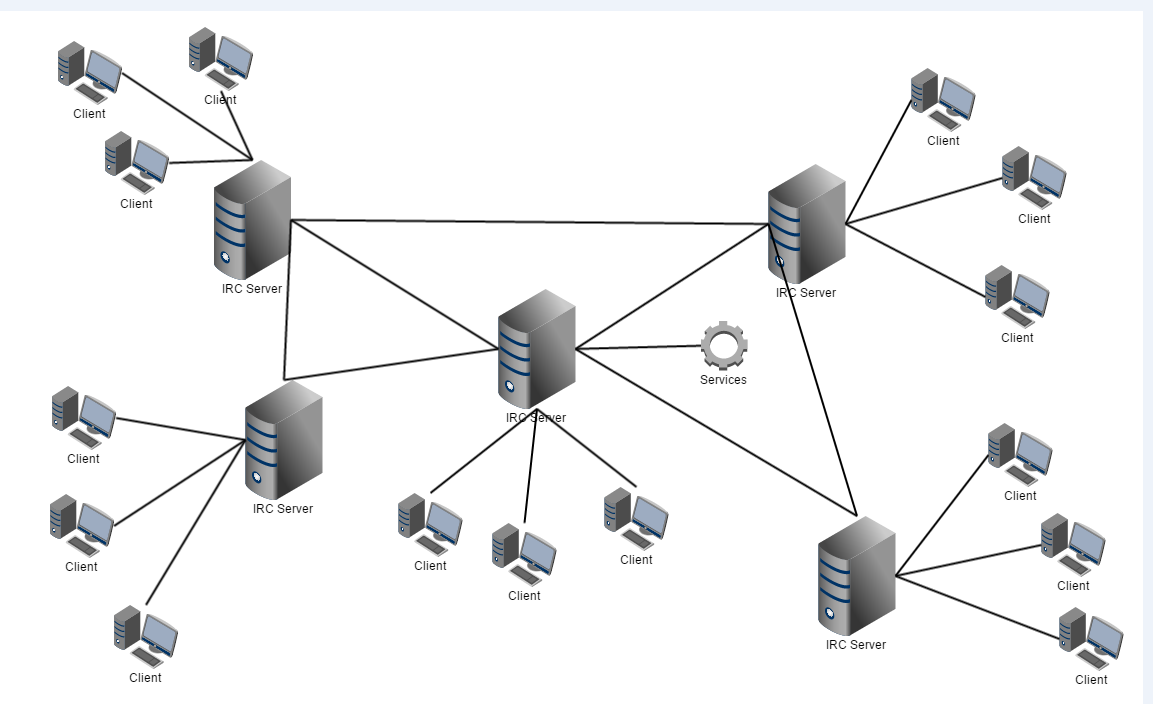


Abb. 2: Schema eines IRC Netzwerkes

Wie in Abb. 2 zu sehen ist, verbinden sich die Clients jeweils zu einem Server. Die Server schliessen sich zu einem Netzwerk zusammen. Jeder Benutzer kann so alle anderen Benutzer erreichen. Das Gleiche gilt für die Services. Die Services sind im Prinzip nichts anderes als ein weiterer Server, der sich mit einem der im sich Netzwerk befindlichen Servern verbindet.

Damit ein Benutzer chatten kann benötigt er einen Client. Diese sind in der Regel kostenlos erhältlich. Populäre Clients sind z.B. mIRC für Windows oder XChat für Linux. Das eigentliche Chatten geschieht über das Senden von Befehlen. Möchte ein Benutzer eine Nachricht an einen anderen Benutzer schicken, so tippt er z.B. /msg Benutzer Nachricht

Der Slash (/) signalisiert dem Client so, dass ein Befehl abgesetzt werden soll. Er wandelt den erhaltenen Befehl dann in das Format um, das vom Protokoll verlangt wird. Die genaue Syntax für die Eingabe kann von Client zu Client variieren, die endgültige Form wird jedoch vom Protokoll definiert.

## 2.3 Protokoll

Um eine einheitliche Form der Nachrichtenübermittlung zu gewährleisten sind die IRC Befehle in einem Protokoll genau definiert.

Somit wird sichergestellt, dass die Nachrichtenübermittlung mit jedem Client und jedem Server in einer strukturierten Art und Weise zuverlässig funktioniert.

Das Protokoll für IRC ist als RFC 1459 (Request for Comments) definiert. Mit einem RFC gibt man ein Dokument für eine Diskussion frei. Es behält den Namen RFC auch dann, wenn es allgemeine Akzeptanz erreicht hat.

Nachfolgend wollen wir einige wichtige Punkte des Protokolls betrachten.

### 2.3.1 Einführung

#### 2.3.1.1 Grundlagen

Das Protokoll legt fest, dass jegliche Nachrichten die über das Netzwerk gesendet werden, aus den drei Teilen

* Präfix (optional)
* Befehl
* Parameter

bestehen und folgendes Format aufweisen müssen:

:Präfix Befehl Argument CRLF

Zwischen dem Kolon (:) und dem Präfix darf kein Leerzeichen stehen und jeder Befehl muss mit Carriage Return (CR) und Line Feed (LF) abgeschlossen werden.

Der Befehl muss ein gültiger IRC Befehl oder dessen numerische Repräsentation sein. Eine Liste gültiger Befehle findet sich im Anhang.

Wir dieses Format nicht eingehalten, wird der Befehl vom Server ignoriert. Auf die meisten Befehle soll der Server eine Antwort senden. Diese hat eine eindeutige Nummer und ist im Kapitel 6 des Protokolls definiert.

#### 2.3.1.2 Operatoren

Um eine grundlegende Ordnung aufrecht zu erhalten kann der Betreiber des IRC Servers sogenannte Operatoren ernennen. Die haben im Vergleich zu anderen Benutzern erweiterte Rechte. Diese sind nicht mit Operatoren in Chaträumen zu verwechseln, welche Störenfriede aus dem Chatraum verbannen können. Vielmehr haben diese Operatoren Rechte die für den ganzen Server gelten. So können diese zum Beispiel einen Benutzer von gesamten Netzwerk verbannen.

#### 2.3.1.3 Channels

Ein Channel ist ein Chatraum. Also ein Gebilde in dem sich mehrere Benutzer treffen und austauschen können. Nachrichten die an einen Channel gesendet werden können von allen Benutzern in diesem Channel gelesen werden. Einen Channel kann man mit dem Befehl JOIN betreten.

Wie in Kapitel 2.3.1.2 erwähnt gibt es neben den Server-Operatoren auch Channel-Operatoren die unerwünschte Benutzer aus dem Channel verbannen können. Diese Operatoren haben auch weitere Rechte, zum Beispiel können diese auch weitere Benutzer zu Operatoren ernennen.

### 2.3.2 Die IRC Spezifikation

Für IRC wurde kein spezifischer Zeichensatz definiert. Vorgegeben ist lediglich, dass ein 8-Bit Zeichensatz verwendet muss. Jedoch hat sich das amerikanische ASCII als Standard durchgesetzt. Das heisst, dass in der Regel keine Umlauft für Server Befehle verwendet können. Dasselbe gilt für Benutzernamen, sogenannte Nicknames.

Die Kommunikation zwischen den Benutzer wird hingegen nicht beeinträchtigt. Dort können jedwede Art von Zeichen gesendet werden, da die meisten Server für die Kommunikation UTF-8 unterstützen.

### 2.3.3 IRC Konzepte

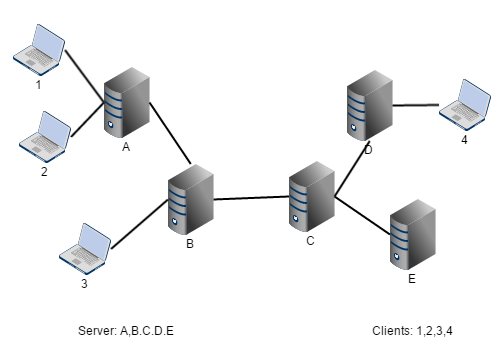
In diesem Kapitel untersuchen wir die protokollarischen Konzepte der Nachrichtenübermittlung basierend auf dem nachfolgenden Bild.

Abb. 3: Struktur eines IRC Netzwerks

#### 2.3.3.1 1:1 Kommunikation

Die Kommunikation auf einer 1:1 Basis erfolgt in der Regel zwischen Benutzern, da die Server nicht ausschliesslich untereinander kommunizieren.

Der Pfad einer Nachricht nimmt stets den kürzesten zwischen zwei Punkten im Netzwerk. 1:1 Nachrichten werden nur von Sender, Empfänger und den Servern, die die Nachricht weiterleiten sollen gesehen.

Beispiel 1: Eine Nachricht zwischen Client 1 und 2 wird nur vom Server A gesehen, der diese direkt an Client 2 schickt.

Beispiel 2: Eine Nachricht zwischen Client 1 und 3 wird von den Server A und B gesehen.

Beispiel 3: Eine Nachricht zwischen Client 2 und 4 wird von den Servern A, B, C und D gesehen.

#### 2.3.3.2 1:n Kommunikation

Die 1:n Kommunikation auf dem IRC tritt in verschiedenen Formen auf. Diese sind:

* Nachricht an eine Liste: Der Client schickt die Nachricht an eine Liste von Empfängern.
* Nachricht an eine Gruppe: Der Client schickt eine Nachricht einen Channel
* Nachricht an einen Host: Der Client schickt einen Nachricht Empfänger welche über die gleiche Hostmaske verfügen. Beispiel: Eine Nachricht an \*@\*.swisscom.com empfangen alle, die über einen Swisscom-Anschluss auf dem Server verbunden sind. Diese Art der Nachricht ist üblicherweise Operatoren vorbehalten.

#### 2.3.3.3 1:Alle Kommunikation

Diese Art der Kommunikation beinhaltet Nachrichten die an alle Clients oder Server oder beides verschickt werden. Dies kann ein hohes Datenvolumen verursachen.

### 2.3.3 Nachrichten Details

Damit sich ein Client zu einer Server verbinden kann, muss sich der Client nach Erstellung der Verbindung mit dem Server registrieren. Dies geschieht über die Übermittlung von Verbindungsnachrichten. Die nachfolgenden Nachrichten müssen demnach gesendet werden.

#### 2.3.3.1 Pass Nachricht

Diese Nachricht ist nur dann erforderlich, wenn der Server mit einem Passwort geschützt ist. Folgende Syntax ist zwingend:

PASS <Passwort> CRLF

#### 2.3.3.2 Nick Nachricht

Mit diesem Befehl legt man seinen eigenen Nicknamen fest.

NICK <Nickname> CRLF

Wird der Nickname bereits verwendet meldet der Server die Fehlermeldung

ERR\_NICKNAMEINUSE

zurück.

#### 2.3.3.3 User Nachricht

Mit dieser Nachricht werden die Details des Client an den Server gemeldet.

USER <Benutzername> <Hostname> <Servername> <Realname>

Der Benutzername ist Teil der Hostmaske. Diese wird in den Einstellungen im Programm zum Chatten in der Regel als E-Mail Adresse angegeben und dem Server so übermittelt.

Der Hostname wird durch den Internetanbieter bestimmt. Er ist von der IP Adresse abstrahiert.

Der Servername wird automatisch gesetzt. Verbindet man sich auf ein Netzwerk mit mehreren Servern ist hier der Name des Servers über den man sich verbindet gemeint.

Der Realname kann frei festgelegt werden. Er kann öffentlich eingesehen werden, daher sollte man sich gut überlegen, ob man seinen echten Namen angeben will.

Als Beispiel können wir den Windows Client mIRC hinzuziehen. Bevor man sich auf einen Chat verbindet muss man in den Einstellungen diese Informationen hinterlegen:

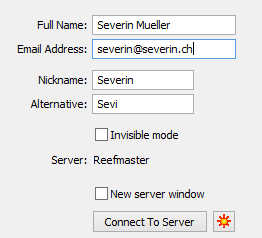


Abb. 4: Verbindungseinstellungen mIRC

Der Full Name ist der Realname und die E-Mail Adresse liefert den Benutzername. Jedoch hat nur der Teil vor dem @ einen Einfluss.

Wie erwähnt können wir den Hostnamen nicht beeinflussen. Wenn uns diese Information interessieren sollte können wir nach dem Verbinden eine WHOIS Nachricht an den Server schicken. Mit WHOIS werden diverse Information über den angegebenen Benutzer angezeigt. Syntax

WHOIS <Nickname>

Wenn wir also im Beispiel Severin den Hostnamen herausfinden möchten, senden wir die Nachricht

/whois Severin

an den Server. Folgendes kommt zurück:

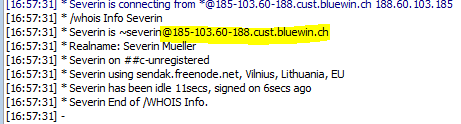


Abb. 5:Rückgabe einer WHOIS Nachricht

Der Hostname ist hier gelb markiert.

#### 2.3.3.4 Server Nachricht

Wollen wir statt einen Client einen Server verbinden, so müssen wir eine Server Nachricht schicken.

SERVER <Servername> <Hopcount> <Info> CRLF

Servername ist der Name des Servers den wir mit dem Netzwerk verbinden wollen.

Hopcount ist eine interne Information die indiziert, wieweit die Server voneinander weg sind (wie viele Knoten dazwischen liegen)

Info ist eine Beschreibung des Servers.

Auf weitere Details des Protokolls verzichten wir an dieser Stelle. In den späteren Kapitels dieser Arbeit werden wir auf die benötigten Teile des Protokolls eingehen.

## 2.4 Services

Wie in Kapitel 1 erwähnt bietet das IRC Protokoll an sich keinerlei Funktionen um einen Nicknamen oder Channel zu reservieren. Für diesen Zweck wurden Services geschaffen.

Nachfolgend möchten wir erläutern, was genau wir unter diesen Services verstehen.

### 2.4.1 Einführung

Services sind im Grunde nichts anderes als ein Server der sich zum IRC Netzwerk verbindet. Auf diesem Server laufen die einzelnen Dienste wie normale Clients, mit dem Unterschied, dass diese eine Kennung besitzen, die sie als Service kennzeichnet. Diese Kennung wird als sogenannter Usermode vom Server zur Verfügung gestellt (siehe Kapitel 4.2.3.2 im IRC Protokoll).

Es gibt keine allgemeingültige Definition welche Services ein Server unterstützen soll. Das Protokoll legt lediglich fest, dass ein Benutzer als Service markiert werden kann.

In der Praxis haben sich jedoch einige Arten von Services durchgesetzt. Dazu zählen:

* Nickserv: Dienst um Nicknames zu registrieren
* Chanserv: Dienst um Channels zu registrieren
* Memoserv: Dienst um Nachrichten an registrierte Benutzer zu senden
* Operserv: Dienst für Server-Operatoren
* Botserv: Dienst für das Erstellen von Bot die als eine Roboter fungieren
* Hostserv: Dienst um für bestimmte Benutzer neue Hosts zu setzen
* Adminserv: Dienst für Server Administratoren

In dieser Arbeit befassen wir uns nur mit Nickserv, Chanserv, Operserv, Botserv und Adminserv.

### 2.4.2 Nickserv

Wie erwähnt stellt Nickserv einen Dienst zur Verfügung mit dem man sich einen gewünschten Benutzernamen registrieren kann. Üblicherweise wird dieser mit einem persönlichen Passwort geschützt. Da heisst, dass man sich für diesen Nick identifizieren muss, bevor man ihn verwenden kann. In der Regel werden weitere Funktionen zur Verfügung gestellt. Meist sind dies Optionen die mit anderen Diensten interagieren.

### 2.4.3 Chanserv

Chanserv oder Channel Services sind ein Dienst für die Registrierung von Channels. Mit einer Registrierung soll sichergestellt werden, dass unbefugte Benutzer den Channel nicht „kapern“ können. Unter kapern verstehen wir das erschleichen von Op-Rechten um dann den Channel zu übernehmen. Da das IRC Protokoll vorsieht, dass jeder Benutzer einem anderen Benutzer Op-Rechte geben kann kam dies häufig vor. Mittels einer Registrierung kann sich der Eigentümer des Channels jederzeit über Chanserv selber die Op-Rechte zurückgeben lassen und so sicherstellen, dass der Channel nicht gekapert wird.

### 2.4.4 Operserv

Operserv soll Server Operatoren die Möglichkeit geben, einige Operation global, also auf allen Server des Netzwerkes zu steuern. Wir wollen einen Schritt weitergehen und Administratoren die Möglichkeit geben, normalen Benutzern ohne Operatoren Recht Zugriff auf Operserv zu erlauben. Dies kann z.B. nützlich sein, wenn kein Operator anwesend ist und man einen Benutzer dem man vertraut die Recht geben kann damit dieser vorläufig nach dem Rechten sieht.

### 2.4.5 Botserv

Es kann vorkommen, dass auf gewissen Netzwerk Channels die nicht registriert wurden eine Möglichkeit geben möchte, ein Takeover (kapern) des Channels zu verhindern, ohne dass der Channel registriert werden muss. Mit Botserv kann man sogenannte Bots erstellen, die die Aufgabe haben, einen Channel offenzuhalten und ein Takeover zu verhindern. Ein Bot wird mit einem Passwort versehen, mit welchen man sich für den Bot identifizieren kann und so auf die Funktionen Zugriff erhält.

### 2.4.6. Adminserv

Mit Adminserv kann man Administratoren erlauben, gewisse Server Operationen durchzuführen. Diese können z.B. ein Neustart der Services, das Speichern der Datenbanken oder das Abschalten der Dienste sein.

Die genaue Implementierung kann sich zwischen den einzelnen Lösungen stark unterscheiden. Wir möchten daher untersuchen, welche Lösungen bereits existieren wo es Lücken gibt, die es zu schliessen gilt.

# 3 Marktanalyse

Wie in der Computerwelt üblich gibt es auch für IRC Services und Server verschiedene Lösungen. Einige davon sind proprietäre, andere Open-Source Lösungen. Wir möchten in diesem Kapitel einige davon genauer betrachten und feststellen, wo unsere Lösung bestimmte Lücken füllen kann.

## 3.1 Webmaster ConferenceRoom

ConferenceRoom von Webmaster ist eine propietäre Lösung die eine All-in-One Lösung bietet. Das heisst, dass der IRC Server und die Services in einem Produkt enthalten sind.

Der Vorteil dieser Lösung liegt in den integrierten Services und der Stabilität der Software.

Der Nachteil sind die eher hohen Kosten. Eine Version für 1000 Benutzer kostet $250.00 und die Version für 10000 Benutzer kostet $995.004

Die 1000 Benutzer Version bietet zudem nur die Dienste Nickserv und Chanserv an. Möchte man Memoserv dazu haben benötigt man die teurere Version.

Der Chat von Bluewin wurde auf einem Conference Room Server betrieben.

## 3.2 Unreal Ircd

Unreal Ircd ist ein Open Source IRC Server der zu den populärsten Servern überhaupt gehört. Er läuft äussert Stabil, bietet eine Vielzahl an Konfigurationsmöglichkeiten und ist aussergewöhnlich gut dokumentiert. Als Open-Source Lösung ist der Server kostenlos erhältlich. Wenn man den Server jedoch modifiziert darf man keine technische Unterstützung erwartet.

Ein Nachteil dieser Lösung ist, dass keine Services mitgeliefert werden. Diesem Problem kann man begegnen indem man auf eine externe Lösung zurückgreift. Nachfolgend betrachten wir einige Open-Source Services.

## 3.3 IRC Services

Die IRC Services sind quasi das Original. Praktisch alle populären Lösungen basieren auf dem original Source Code von Andrew Church. Die erste Version wurde 1996 entwickelt. Diese Service werden nicht mehr weiterentwickelt. Die allererste Version dieser Lösung ist immer noch online auf <http://achurch.org/services/> zu finden.

Da die Entwicklung vor einiger Zeit eingestellt wurde bieten diese Services nicht den Umfang die moderne Systeme bieten.

## 3.4. Anope Services

Anope ist eine Services Lösung die auf den originalen IRC Services basiert. Anope bietet eine Vielzahl an Optionen und läuft äusserst stabil. Anope verfügt über eine gute Dokumentation und 6 Services. Zudem werden Module, die einfach dazu geschaltet werden können, unterstützt.

Der Funktionsumfang von Anope ist beträchtlich.

Ein Nachteil ist jedoch, dass sich die Syntax und die Ausgaben teilweise stark von denen die ConferenceRoom bieten unterscheiden und ehemalige Swisscom-Chatter Mühe haben, sich zurecht zu finden.

## 3.5 Auspice

Auspice sind eine Open-Source Lösung die auch unter Windows funktioniert. Leider wurde die Entwicklung 2005 eingestellt. Sie sind aber nach wie vor zum Download erhältlich und auch in Betrieb.

## 3.6 Fazit

An den heute erhältlichen IRC Services ist zu sehen, dass IRC an Popularität eingebüsst hat. ConferenceRoom und Anope werden weiterentwickelt, genau wie auch Unreal IRCd, die weiteren Services werden nicht mehr wirklich gepflegt.

Da viele der ehemaligen Swisscom Chatter sich wünschen, dass die Services ähnlich aufgebaut sind wie diejenigen von ConferenceRoom soll unsere Lösung dieses Ziel verfolgen.

# 4 Anforderungen und Architektur

In diesem Kapitel wollen beginnen, uns mit der Umsetzung zu beschäftigen. Zunächst wollen wir die Use Cases beschreiben. Aufgrund der Use Cases erstellen wir die Anforderungen. Als Akteure sollen die Stakeholder dienen die wir wie folgt definieren.

## 4.1 Basis-Server

Wir definieren die Stakeholder für den Basis-Server wie folgt:

|  |  |
| --- | --- |
| **Stakeholder** | **Beschreibung** |
| Server Administrator | Kann den Server konfigurieren und verfügt über alle Rechte |

Tabelle 1: Stakeholder

Als weiteren Akteur definieren wir:

|  |  |
| --- | --- |
| **Akteur** | **Beschreibung** |
| Basis-Server | Der Basis auf dem die Services laufen. Verbindet sich zum IRC Server |

Tabelle 2: Akteure Basis-Server

Als Basis Server wird der eigentlich Server bezeichnet, auf dem die Services laufen sollen. Der Server soll konfigurierbar sein, demnach muss das Programm einen Mechanismus zur Konfiguration bereitstellen.

### 4.1.1 Use Case Basis Server (UC S-01)

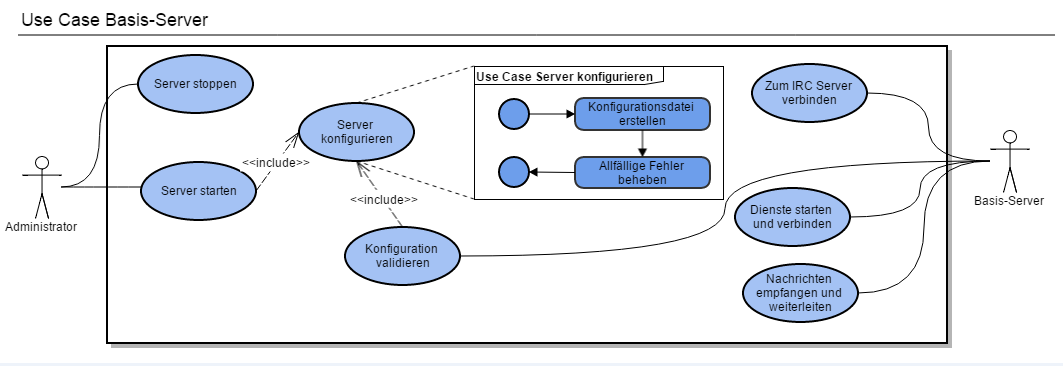


Abb. 6: Use Case Basis Server

Die Benutzer schicken ihre Nachrichten an den IRC Server und nicht an den Services-Server. Der IRC Server leitet dann die Nachricht an den richtigen Service weiter.

### 4.1.2 Anforderungen Basis-Server

#### 4.1.2.1 IRC Server

Zunächst müssen wir festlegen, welche Chat-Server überhaupt unterstützt werden sollen. Die Entscheidung fiel auf den Unreal IRCd Server, da dieser der populärste ist und über eine gute dokumentierte Unterstützung für Services-Schnittstellen verfügt.

Daraus resultiert folgende Anforderung:

|  |  |
| --- | --- |
| **Requirement Nr.** | **R-S-001** |
| Titel | Unterstützte IRC Server |
| Beschreibung | Der Unreal IRCd Server soll unsere Services unterstützen. Es sind die Grundlagen zu schaffen, dass der Basis-Server sich zu einem bestehenden Unreal IRCd Server verbinden kann. |
| Begründung | Unreal IRCd ist der populärste Open-Source IRC Server. So werden die Services einem breiten Publikum angeboten |
| Use Case | UC S-01 |
| Benutzerzufriedenheit | Kein Einfluss |
| Kommentare | - |

Tabelle 3: R-S-001: Unterstützte IRC Server

#### 4.1.2.2 Konfiguration Basis-Server

Die Services sollen einen hohen Grad an Flexibilität aufweisen. Daher ist es erforderlich, dass alle wichtigen Werte des Basis-Servers konfiguriert werden können.

|  |  |
| --- | --- |
| **Requirement Nr.** | **R-S-002** |
| Titel | Konfiguration Basis-Server |
| Beschreibung | Die Grundeinstellungen des Basis-Server sollen konfigurierbar sein. Namentlich sind dies:   * Name des Basis-Servers * Beschreibung * Username * Hostname * Post * Adresse des IRC Servers * Name des IRC Servers   Die Konfiguration soll als Text-Datei vorliegen. |
| Begründung | Unreal IRCd verlangt die Angabe von Name, Username und Hostname sowie den Port. Die Beschreibung dient zu Informationszwecken. Die Adresse des Zielserver ist erforderlich, damit die Services wissen, wohin sie sich verbinden müssen. |
| Use Case | UC S-01 |
| Benutzerzufriedenheit | Kein Einfluss |
| Kommentare | - |

Tabelle 4: R-S-002: Konfiguration Basis-Server

|  |  |
| --- | --- |
| **Requirement Nr.** | **R-S-003** |
| Titel | Starten des Servers |
| Beschreibung | Die Services sollen von der Kommandozeile gestartet werden können. Als Argument für den Programmstart wird „start“ festgelegt. |
| Begründung | Unreal IRCd ist der populärste Open-Source IRC Server. So werden die Services einem breiten Publikum angeboten |
| Use Case | UC S-01 |
| Benutzerzufriedenheit | Kein Einfluss |
| Kommentare | - |

Quellen

1 <http://www.selflinux.org/selflinux/html/irc_geschichte01.html> abgerufen 01.01.2015

2 <https://de.wikipedia.org/wiki/Internet_Relay_Chat#Entwicklung>, abgerufen 01.01.2015

3 <http://www.at-mix.de/internet/internet-0207.htm>, abgerufen am 01.01.2015

4 http://www.webmaster.com/crtabeditions.htm , aberufen am 02.01.2015