h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

– Fachbereich Informatik –

Ausfallsicherheits- und Lastverteilungskonzept

für eine Voice-over-IP-Nebenstellenanlage

Abschlussarbeit zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt von

Sven Eisenhauer

Referent: Prof. Dr. Klaus Frank

Korrefent: Prof. Dr. Klaus Wentzel

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen und Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den

Sven Eisenhauer

Abstract

In dieser Arbeit wird ein Konzept zur Ausfallsicherung einer VoIP-Telefonanlage (PBX) beschrieben und prototypisch umgesetzt. Die erarbeitete Lösung basiert vollständig auf Open-Source-Software und zeigt den Weg von den Grundlagen der VoIP-Technologie und der PBX-Software Asterisk hin zu einem Hochverfügbarkeits(HA)-Cluster. Realisiert wird das HA-Cluster aus zwei Knoten auf Basis der Shared Storage Lösung DRBD und der Redundanz-Lösung Heartbeat.
Als Grundlage des entwickelten Ausfallsicherheitskonzepts dient eine Risiko-Analyse für den allgemeinen Betrieb von IT-System sowie im speziellen für den Betrieb von softwarebasierten TK-Anlagen. Um den speziellen Herausforderungen der Redundanz von TK-Komponenten Rechnung zu tragen, werden verschiedene Hard- und Software-Systeme vorgestellt, um POTS-, TDM- und GSM-Verbindungen in einem Hochverfügbarkeits-Cluster nutzen zu können. Ebenso beinhaltet die Arbeit einen kurzen Überblick über die verschiedenen Kategorien von Clustern. Das Spektrum reicht von Hochverfügbarkeits-Clustern über Hochgeschwindigkeits-Cluster bis hin zu Grids.
Ergänzend wird die Erweiterung des Grundkonzepts um die aktive Nutzung aller Cluster-Knoten zur Lastverteilung untersucht. Auf Basis der Load Balancer Lösung ldirectord werden Client-Anfragen auf die verfügbaren Knoten verteilt. Der gemeinsame Zugriff auf den Shared Storage wird mittels des Cluster-Dateisystems OCFS2 umgesetzt. In dieser Arbeit findet der Ansatz durch die in dem Telefonanlagenprodukt verwendete Datenbanksoftware MySQL aber seine Grenzen. MySQL sieht einen properitären Weg zur Cluster-Bildung seiner Datenbank-Instanzen vor. Dieser ist leider mit dem angestrebten HA-Clusterkonzept aus zwei Knoten nicht kompatibel. Ein theoretischer Lösungsweg für mehr als zwei Knoten wird jedoch aufgezeigt.

Vorwort

Die Motivation für diese Arbeit ergibt sich dadurch, dass die Telefonie für viele Unternehmen und Institutionen einen kritischen Geschäftsbereich darstellt. Mit der zunehmenden Verbreitung der VoIP-Technologie und der damit einhergehenden Ablösung klassischer Telefonie-Technologien wie ISDN, proprietärer Lösungen einiger Hersteller und analoger Systeme, ergibt sich die Frage nach der Zuverlässigkeit von VoIP-Lösungen.

Für viele Unternehmen wäre ein Ausfall der Telefonie-Infrastruktur direkt und indirekt mit hohen Kosten verbunden. Diese Kosten entstehen beispielsweise durch unterbrochene Geschäftsprozesse, die an bestimmten Stellen die Telekommunikation mit einbeziehen oder durch einen Imageverlust des Unternehmens in der Öffentlichkeit, wenn es telefonisch nicht erreichbar ist.

Im Bereich öffentlicher Institutionen wie beispielsweise der Polizei oder bei Rettungsleitstellen wird der Bedarf an zuverlässiger Telekommunikation besonders deutlich.

Die Zuverlässigkeit bestehender VoIP-Lösungen ist aus diesen Gründen von besonderer Bedeutung. Dies belegen auch Studien, die sich mit den Gründen für die schleppende Einführung von VoIP-Technologien in Institutionen beschäftigen.

Den Ausgangspunkt für diese Arbeit stellt das Produkt independent der independent GmbH dar. Dabei handelt es sich um eine Softwarelösung zur Realisierung von Nebenstellenanlagen auf der Basis von VoIP-Technologien, im Folgenden auch als VoIP-Software-PBX bezeichnet. Independent basiert auf der Server-Variante der Linux-Distribution Ubuntu in der Version 6.06 und verwendet Open-Source-Komponenten zur Realisierung der einzelnen Dienste. So umfasst independent nicht nur die reine VoIP-Komponente, sondern unter anderem auch noch ein Web-Interface zur Administration und einen E-Mail-Server für den Versand von Benachrichtigungen

an die Anwender. Aus diesem Kontext ergibt sich die zentrale Fragestellung für die Produktentwicklung. Wie könnte eine Erweiterung des Produkts independent in Richtung Ausfallsicherheit und Lastverteilung aussehen?
Gegenstand dieser Arbeit ist ein Überblick über verschiedene Alternativen und eine prototypische Umsetzung einzelner Lösungsansätze.
An dieser Stelle möchte ich mich bei den zahlreichen Personen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben. An erster Stelle bedanke ich mich bei meinem Referenten für diese Arbeit Herrn Prof. Dr. Klaus Frank und dem Korreferenten Prof. Dr. Christoph Wentzel für Ihre Unterstützung. Weiterhin bedanke ich mich bei meinem Betreuer bei der independent GmbH, Herrn Benjamin Heckmann, M. Sc. für seinen fachkundigen Rat und sein Engagement. Auch den Gesellschaftern der independent GmbH, namentlich Herr Prof. Dr. Klaus W. Wente, Prof. Dr. Udo Bleimann, Prof. Dr. Günter Turetschek, Herr Harald Zapp und Herr Busch, schulde ich meinen Dank, denn ohne ihr Engagement wäre independent und somit auch diese Arbeit nicht möglich gewesen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Diese Arbeit zeigt, wie mit kostengünstiger Hardware und Open-Source-Software ein ausfallsicheres und performantes Cluster für VoIP-Systeme realisiert werden kann. Das Hauptziel dabei ist es, die Verfügbarkeit der Dienste im Vergleich zu einem Einzelsystem zu erhöhen. Ein weiterer Aspekt dieser Arbeit befasst sich damit, wie Reaktionszeit der Dienste gering und die Qualität hoch gehalten werden kann.
Nach einer Präsentation der notwendigen Grundlagen für die hier erarbeiteten Konzepte in Kapitel 2, erfolgt in Kapitel 3 die Beschreibung einer VoIP-Software-PBX mit erhöhter Ausfallsicherheit.
Ausgehend von einer Risiko-Analyse, welche Komponenten eines VoIP-Systems bei einem Ausfall die Verfügbarkeit der Dienste beeinträchtigen, werden Lösungsansätze für die Erhöhung der Zuverlässigkeit der jeweiligen Komponenten aufgezeigt. Dabei liegt der Schwerpunkt darauf, die für den Anwender sichtbaren Ressourcen auf den einzelnen Cluster-Knoten zu platzieren. In Kapitel 4 befasst sich die Arbeit mit der Konsistenz gemeinsamer Daten auf den einzelnen Cluster-Rechnern.
Auf diesen Ergebnissen aufbauend, wird in Kapitel 5 ein Konzept erarbeitet, wie die relevante Last auf dem Cluster verteilt werden kann. Zusätzlich enthält es eine Darstellung der bei der Implementierung aufgetretenen Probleme.
Das in Kapitel 6 vorgestellte Ergebnis der Arbeit ist ein ausfallsicheres Cluster. Zusätzlich zeichnet sich die Lösung dadurch aus, dass sie Client-unabhängig ist und rein Server-seitig realisiert wird.
In Kapitel 7 wird abschließend ein Ausblick auf Themen gegeben, die auf Basis dieser Arbeit weiter bearbeitet werden können. Besonderes Augenmerk ist hierbei auf die Erarbeitung von Lösungsansätzen für die in Kapitel 5 festgestellten Problematiken zu legen.

2 Grundlagen

2.1 VoIP

Voice-over-Internet-Protocol (abgekürzt VoIP) oder auch IP-Telefonie bezeichnet die Übertragung von Telefongesprächen über IP-Netze. Im Gegensatz zu herkömmlichen Telefonverbindungen, die leitungs- bzw. vermittlungsorientierte Punkt-zu-Punkt-Verbindungen darstellen, handelt es sich bei IP-Netzen um paketorientierte Netze. Diesem besonderen Umstand muss bei der Betrachtung von VoIP stets Rechnung getragen werden. IP-Netze können verlustbehaftet sein und die Laufzeit von Paketen kann nicht garantiert werden.

Ein Telefongespräch über eine IP-Infrastruktur umfasst zwei Aspekte. Zum Einen die Übermittlung von Steuerdaten, wie z. B. die Signalisierung eines Rufaufbaus, und zum Anderen die Übertragung der Sprache. Bei der menschlichen Sprache handelt es sich um akustische Schwingungen im Frequenzbereich von ca. 100 Hz bei einem Mann und ca. 200 Hz bei einer Frau bis ca. 3,4 kHz. Für die Sprachübertragung in Telefonnetzen wird der Frequenzbereich von 300 Hz bis 3,4 kHz genutzt, woraus sich eine Bandbreite von 3,1 kHz ergibt. Bei der analogen Telefonie werden diese akustischen Schwingungen von einem Mikrofon aufgezeichnet und in elektrische Schwingungen umgewandelt. Diese elektrischen Schwingungen werden über das Telefonnetz zum Empfänger übertragen und dort von einem Lautsprecher wieder in akustische Schwingungen umgesetzt. Für die digitale Übertragung von Sprache, sowohl beim Integrated Services Digital Network (abgekürzt ISDN) als auch bei VoIP, müssen die analogen Schwingungen in diskrete Werte umgewandelt werden. Dazu gibt es unterschiedliche Verfahren, die sich in Abtast- und Bitrate unterscheiden. Dies ergibt Unterschiede beim Berechnungsaufwand der Codierung bzw. Decodierung, beim Bandbreitenbedarf und in der Sprachqualität. Alle basieren aber auf dem Nyquist-

Shannon-Abtasttheorem, das stark vereinfacht aussagt, dass die Abtastfrequenz bei der Digitalisierung größer als die zweifache Bandbreite sein muss. Das ISDN verwendet eine Abtastrate von 8 kHz (entspricht näherungsweise 2 * 3,1 kHz) bei einer 8 bit Abtastung. Daraus ergibt sich ein Bandbreitenbedarf von 64kbit pro Sekunde (8000 * 8 bit), was genau der Bandbreite eines ISDN B-Kanals entspricht. Ein ISDN-Standardanschluss verfügt über zwei B-Kanäle, auch als Nutzkanäle bezeichnet, zur Übertragung von Sprache, Fax oder Daten und einem D-Kanal(16 kbit/s Bandbreite) zur Übertragung von Signalisierungsdaten.

Auch bei VoIP werden entsprechende Digitalisierungsverfahren, sog. Codecs (Coder-Decoder), verwendet, um die Sprache zu codieren bzw. zu decodieren.

Nach dem OSI-Modell werden die digitalisierten Sprachdaten mit einem Layer 7 Protokoll, wie dem Realtime Transfer Protocol (abgekürzt RTP), zwischen den Endgeräten übertragen. Eine solche Übertragung wird in diesem Zusammenhang oft als Medienverbindung oder auch Medienkanal bezeichnet.

Auch bei der Signalisierung kommen unterschiedliche Verfahren zum Einsatz. Bei herkömmlichen Nebenstellenanlagen verwendet fast jeder größere Hersteller eigene Lösungen, die nicht mit anderen Systemen kompatibel sind. Ebenso finden unter dem Oberbegriff VoIP viele verschiedene Lösungen Verwendung. Neben hersteller-eigenen Protokollen existieren einige standardisierte Protokolle.

Der folgende Abschnitt stellt die wichtigsten offenen Signalisierungsprotokolle für VoIP vor.

Das zur Zeit am besten unterstützte und am verbreitetste Protokoll ist das Session Initiation Protocol (abgekürzt SIP). Es existiert eine Vielzahl von Endgeräten, die dieses Protokoll unterstützen. Diese Endgeräte sind in Form von Softphones, Hardphones und Adaptern am Markt in großer Auswahl erhältlich. Bei Softphones handelt es sich um Programme, die die VoIP-Funktionen realisieren. Die Sprachaufnahme bzw. -wiedergabe erfolgt meist über ein Headset. Im Gegensatz dazu

handelt es sich bei einem Hardphone um ein eigenständiges Gerät, das in Aussehen und Handhabung einem herkömmlichen Telefon ähnelt. Statt an einen analogen oder ISDN-Anschluss wird dieses Gerät an ein IP-Netz angeschlossen. Ein SIP-Adapter, auch ATA genannt, verbindet ein VoIP-Netzwerk mit einem herkömmlichen Endgerät. Dadurch kann ein bereits vorhandenes Gerät auch in einem VoIP-Netzwerk verwendet werden.

SIP arbeitet transaktionsorientiert, ähnlich HTTP, und stellt verschiedene Methoden bereit, die durch entsprechende Anfragen (sog. Requests) ausgelöst werden. Die Tabelle SIP Methoden [ACK-2007] liefert einen Überblick über die wichtigsten SIP Methoden.

Methode	Bedeutung
REGISTER	Registrierung der Adresse und der Ports, unter der ein Teilnehmer erreichbar ist, zusätzlich können auch so genannte CPL-Skripte (Call Processing Language) transportiert werden
INVITE	Auslösung eines Anrufs, Transport der Parameter für die Medienkanäle und Möglichkeit zur Modifikation von Medienverbindungen in einem laufenden Gespräch durh so genannte Re-INVITEs
ACK	Abschließende Nachricht im 3-Wege-Aufbau von Verbindungen
CANCEL	Abbruch eines eingeleiteten Requests
BYE	Beendigung eines Gesprächs
OPTIONS	Abfrage der vom Kommunikationspartner unterstützten Fähigkeiten

Tabelle 1: SIP Methoden [ACK-2007]

Da SIP fortlaufend erweitert und neue Funktionen ergänzt werden, kommen auch ständig neue Methode hinzu.

Gleichzeitig wächst der Markt von Telefongesellschaften, die ihren Kunden SIP-Anschlüsse anbieten, stetig. Neben SIP spielt das Inter Asterisk Exchange Protocol (abgekürzt IAX) eine wichtige Rolle, da diese Protokoll auch vermehrt von Herstellern

und Telefonanbietern eingesetzt und unterstützt wird. Frühe VoIP-Lösungen verwendeten vor der Einführung von SIP und IAX den H.323 Standard, der aufgrund seiner Komplexität nur noch selten Verwendung findet. Aufgrund der hohen Verbreitung von SIP legt diese Arbeit auch ihren Schwerpunkt auf dieses Protokoll. Ein SIP-Netzwerk besteht aus mehreren Komponenten, die jeweils eine oder mehrere Rollen übernehmen. Die Tabelle *SIP Rollen* zeigt eine Übersicht über die verschiedenen Rollen und deren Bedeutung.

Rolle	Erläuterung
SIP User Agent	Endpunkt einer SIP-Verbindung, dies kann ein Hard- oder Softphone, ein Adapater oder ein SIP Server sein.
SIP Proxy Server	Routet SIP-Nachrichten zwischen Endgeräten
SIP Registrar Server	Verwaltet die logischen Informationen der angeschlossenen Endgeräte, wie bespielsweise deren IP-Adresse. Die Aktualisierung der Daten erfolgt dynamisch mittels REGISTER Nachrichten
SIP Redirect Server	Stellt SIP-Routing-Informationen bereit und liefert alternative Routen, falls sich Registrierungen von User Agents ändern

Tabelle 2: SIP Rollen

2.2 VoIP-Nebenstellenanlagen

Eine Nebenstellenanlage (englisch abgekürzt PBX, private branch exchange) verwaltet die Telekommunikation einer Institution oder eines Unternehmens. Diese Arbeit geht von einer Anbindung der Endgeräte mittels VoIP-Technologie aus. Zur Anbindung der Endgeräte wird in den meisten Fällen Ethernet verwendet. Ethernet hat sich in den letzten Jahren als defacto-Standard bei der Verkabelung von Local Area Networks

(abgekürzt LAN) etabliert und ist mittlerweile in den meisten Institutionen anzutreffen. Auch verfügen nahezu alle SIP Endgeräte über einen Ethernet-Anschluss.
Die Anbindung an das öffentliche Fernsprechnetz (englisch abgekürzt PSTN, Public Switched Telephone Network) erfolgt in unterschiedlicher Art und Weise, je nach Anwendungsfall. Dabei stehen verschiedene Alternativen zur Auswahl, die auch beliebig kombinierbar sind. Eine VoIP-PBX kann direkt an das ISDN mit entsprechenden Schnittstellenkarten für die S_0- oder S_2M-Schnittstelle angeschlossen werden. In einigen Ländern, besonders in den USA, ist Verbreitung von ISDN begrenzt. Dort erfolgt eine Anbindung meist über analoge Schnittstellen. Auch eine direkte Anbindung der PBX an Mobilfunknetze unter Verwendung des GSM-Standards ist realisierbar, da auch hierfür entsprechende Schnittstellenkarten auf dem Markt verfügbar sind. Eine weitere Alternative ist die Anbindung der PBX über VoIP-Technologien an einen entsprechenden Anbieter. Dieser stellt in diesem Fall den eigentlichen Übergang in das PSTN bereit und leitet die Verbindungen über IP-Netze zu seinen Kunden. Bei dieser Anbindung ist besonderes Augenmerk auf die Bandbreite des IP-Anschlusses zu legen, um die benötigte Kapazität an parallelen Gespräch bereitstellen zu können.
Im Fall von independent verfügt die PBX über ein datenbank-basiertes Web-Interface zur einfachen Konfiguration des Systems, um dem Endanwender die Handhabung zu erleichtern.

2.3 Asterisk

Bei Asterisk handelt es sich um eine Open-Source-Software zur Realisierung einer PBX und damit die zentrale Komponente von independent. Asterisk wurde ursprünglich für das Betriebssystem Linux entwickelt und läuft dort als Systemdienst. Der grundlegende Ansatz von Asterisk ist es, Verbindungen von einem Channel zu einem anderen Channel herzustellen. Unter einem Channel versteht man dabei im

Asterisk-Kontext eine beliebige Anbindung. Dies können ISDN-Kanäle, SIP-Endgeräte, VoIP-Anbindungen oder andere Objekte sein.
Eine Verbindung wird aus Sicht von Asterisk von einem Channel aufgebaut und in einem Kontext verarbeitet. Die Steuerung der Abläufe in einem Kontext wird durch den so genannten Dialplan festgelegt. Für eine Rufnummer in einem Kontext können mehrere Aktionen nach ihrer Priorität sequentiell von verschiedenen Applications abgearbeitet werden. Unter einer Application ist hierbei eine Asterisk-Funktion zu verstehen. Beispiele für einfache Applications sind Dial oder Hangup, um Verbindungen aufzubauen oder zu beenden. Durch die Vielzahl und Unabhängigkeit der bei Asterisk zur Verfügung stehenden Applications und Channels kann ein Asterisk-System als Medien-Umsetzer bzw. VoIP-Gateway verwendet werden.
Die Konfiguration von Asterisk erfolgt über eine Vielzahl von Konfigurationsdateien, in denen die entsprechenden Konfigurationsparameter der entsprechenden Komponenten hinterlegt sind. Es ist auch möglich, Asterisk dahingehend um zu konfigurieren, Teile seiner Konfiguration aus einer MySQL-Datenbank zu beziehen.
Die Abbildung 1 zeigt einen Überblick über die Architektur des Asterisk-Kerns. Durch diese modulare und flexible Architektur existiert mittlerweile eine unüberschaubare Vielfalt an Asterisk-Erweiterungen für unterschiedlichste Zwecke. Sowohl private Entwickler als auch Software-Firmen zeigen große Aktivität bei der Entwicklung von Asterisk-Erweiterungen.
In den Szenarien, die diese Arbeit abdeckt, kann Asterisk in allen SIP-Rollen Verwendung finden. Er kann für Hard- oder Softphone Registrar-, Proxy-, Redirect- und Gateway-Server darstellen, gleichzeitig kann beispielsweise die VoiceMail-Komponente als User-Agent agieren.

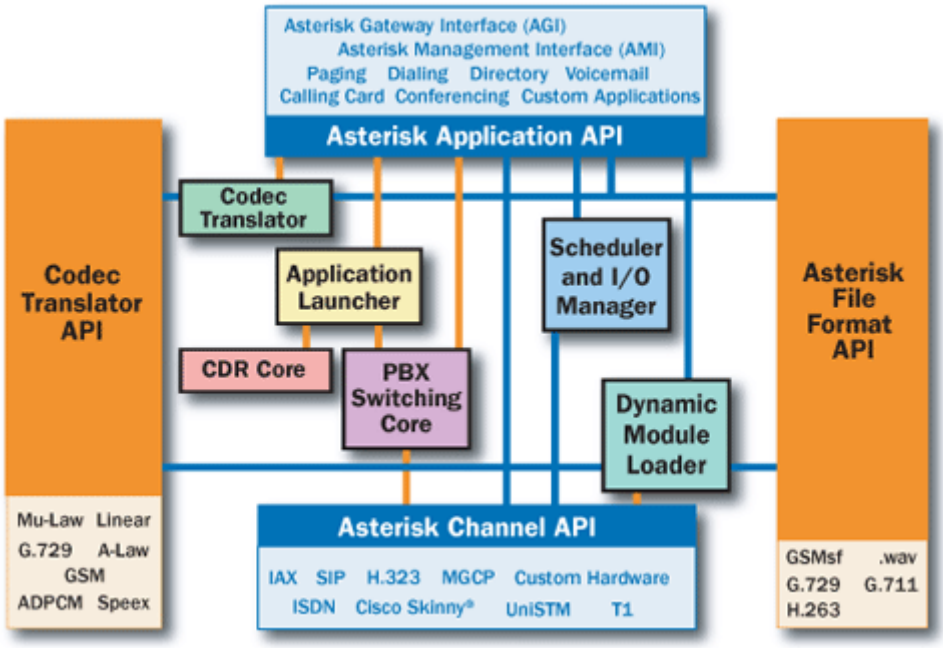

Abbildung 1: Asterisk-Architektur

(Quelle: www.digium.com/images/graphics/asteriskarch.gif)

2.4 Ubuntu Server 6.06 LTS

Bei Ubuntu handelt es sich um eine Distribution des Betriebssystems Linux. Eine Distribution stellt verschiedene Elemente eines Linux-Systems aus einer Quelle und aufeinander abgestimmt bereit. So ist viel verschiedene Software, besonders im Opensource-Bereich aus vielen Quellen und in vielen Versionen verfügbar. Diese vielen unterschiedlichen Versionen und Abhängigkeiten von verschiedenen Software-Komponenten untereinander zeigen deutlich die Vorteile bei der Verwendung einer Distribution im Gegensatz zum Einsatz einzelner unabhängiger Software-Komponenten. Ubuntu bietet vorkompilierte Pakete für nahezu sämtliche in dieser Arbeit verwendete Software. Außerdem bietet es einen langen Support-Zeitraum (LTS, long time support) für die Server Variante. Dieser lange Support-Zeitraum stellt eine interessante Plattform für den Aufbau kommerziell genutzter Systeme bereit. Aus

diesen Gründen versucht diese Arbeit möglichst viele fertige Ubuntu Pakete und Mechanismen zu verwenden. Dies sollte auch eine spätere Integration dieser prototypischen Umsetzung in das Produkt erleichtern.
Zur grundlegenden Installation und Konfiguration des Ubuntu Servers so wie zur Paketverwaltung verweise ich an dieser Stelle auf [USG-2007].

2.5 Cluster

Der Begriff des Clusters findet in mehreren Bedeutungen in der Informatik Verwendung. Im Allgemeinen besteht ein Cluster aus einzelnen eigenständigen Rechnersystemen, den so genannten Knoten bzw. Cluster-Knoten. Die minimale Anzahl Knoten in einem Cluster beträgt zwei. Jeder Knoten übernimmt eine bestimmte Rolle innerhalb des Clusters und es besteht eine Kommunikationsschicht innerhalb des Clusters zwischen den Knoten. Ein Cluster erscheint dem Benutzer bzw. Client wiederum als ein einzelner Computer.

Für alle Cluster-Arten existieren unterschiedliche Lösungen und Produkte diverser Hersteller. Der Fokus dieser Arbeit liegt dabei auf Lösungsansätzen, die auf Open-Source-Software basieren.

Cluster lassen sich in vier Kategorien unterteilen, wobei die Grenzen nicht klar definiert sind bzw. ein Cluster mehreren Kategorien zugeordnet werden kann. Die Unterscheidungsmerkmale der Cluster-Kategorien stellen hauptsächlich die Rollen der einzelnen Knoten und deren Kopplungsgrad dar. Entsprechend [CLU-2007] orientiert sich diese Arbeit an denen im Folgenden erläuterten Kategorien für Cluster.

2.5.1 Hochverfügbarkeits-Cluster

Der Hauptzweck eines Hochverfügbarkeits-Cluster, englisch High-availability-Cluster oder abgekürzt HA-Cluster, besteht darin die Verfügbarkeit und Zuverlässigkeit eines Dienstes zu verbessern, der vom Cluster angeboten wird. Dazu findet mindestens ein

redundanter Knoten Verwendung, um den so genannten Single-Point-Of-Failure (SPOF) zu eliminieren. Das Ziel besteht darin, alle Komponenten redundant auszulegen, sodass kein SPOF mehr existiert. Dadurch hat der Ausfall einer Komponente keinen Einfluss mehr auf die Verfügbarkeit des Dienstes. Im Linux-Umfeld hat sich die Open-Source-Software „heartbeat“ [ROB-2000] des „Linux HA Project“ [ROB-2007] zur Implementierung von HA-Clustern etabliert.

Nach [REI-2000] lässt sich die Verfügbarkeit eines Systems wie folgt berechnen:

$$A = \frac{MTTF}{MTTF + MTTR}$$

Abbildung 2: Berechnung der Verfügbarkeit eines Systems

A bezeichnet hierbei die Verfügbarkeit (availability).

MTTF bezeichnet die durchschnittliche Dauer der fehlerfreien Funktion (mean time to failure).

MTTR bezeichnet die durchschnittliche Dauer einer Reparatur (mean time to repair)

Der Ausdruck MTTF+MTTR wird auch als MTBF (mean time between failure) bezeichnet und drückt die durchschnittliche Zeit zwischen zwei Ausfällen aus.

Allerdings kann diese Betrachtung erst am Ende des Lebenszyklus eines Systems erfolgen. Aus diesem Grund begrenzt man zur Klassifizierung der Verfügbarkeit von Systemen den Betrachtungszeitraum auf ein Jahr. Die Klassengrenzen sind definiert durch die Forderung nach bestimmten maximalen Ausfallzeiten, relativ zum Zeitraum eines Jahres. Daraus resultieren die gezeigten absoluten Zeitangaben.

Verfügbarkeit	Maximale Ausfallzeit	Klasse
99 %	88 h	2
99,9 %	9 h	3
99,99 %	52 min	4
99,999 %	5 min	5
99,999 %	31 sec	6

Tabelle 3: Verfügbarkeitsklassen [REI-2000]

2.5.2 Lastverteilungs-Cluster

Die Haupteigenschaft eines Lastverteilungs-Clusters, oder auch englisch Load-Balancing-Cluster genannt, besteht darin, dass Anfragen an das Cluster auf mehrere Knoten verteilt werden und so die Abarbeitung der Anfragen beschleunigt wird. Die Aufgabe der Verteilung übernimmt ein so genannter Load-Balancer. Auch zur Implementierung dieser Art von Clustern steht Open-Source-Software in Form des „Linux Virtual Server“ [ZHA-1999] ein Load-Balancer zur Verfügung.

Diese Form von Cluster findet besonders stark bei Web-Servern Verwendung.

2.5.3 Hochgeschwindigkeits-Cluster

Hierbei spricht man im Englischen von High-performance computing (HPC) clusters. Diese Cluster sind darauf ausgelegt rechenintensive Aufgaben mit den Mitteln der parallelen Datenverarbeitung möglichst schnell auszuführen. Dabei handelt es sich nicht um eigenständige Anfragen, sondern um Teilaufgaben eines komplexen Gesamtproblems. Jedem Knoten wird dabei ein kleiner Teil des Gesamtproblems zur Verarbeitung übergeben. Laut [BAD-2001] lassen sich große Geschwindigkeitsgewinne durch diese Parallelverarbeitung erreichen. Mit Beowulf [LIN-2005], PVM [SUN-1990] und OpenMPI [GAB-2004] stehen auch hier Open-

Source-Software bzw. offene Standards zur Verfügung. Damit stehen dem Applikationsentwickler Programmiermodelle zur Verfügung, die Schnittstellen zur Parallelisierung auf Anwendungsebene bieten.

2.5.4 Grid-Cluster

Grid-Cluster bzw. Grid-computing unterscheidet sich von den anderen Clustern-Arten in der Vertrauensstellung und der geographischen Lage der Knoten. Während sich die Knoten bei den vorgenannten drei Cluster-Arten im Regelfall geographisch nahe beieinander befinden und über schnelle LAN-Netzwerke verbunden sind, können sich Grid-Knoten durchaus weit voneinander entfernt befinden und über WAN-Leitungen verbunden sein. Die Teilaufgaben, die den einzelnen Grid-Knoten zugewiesen werden, besitzen keine Datenabhängigkeiten untereinander.

3 VoIP-Software-PBX-HA-Cluster

Das folgende Kapitel zeigt, welche Ausfallrisiken bei dem Betrieb einer VoIP-Software-PBX bestehen, nennt unterschiedliche Lösungsansätze und beschreibt ein HA-Konzept für die VoIP-Software-PBX.

3.1 Risiko-Analyse einer VoIP-Software-PBX

Ausgehend von einer VoIP-Software-PBX, die auf einem handelsüblichen Standard-PC als Hardware-Plattform basiert, analysiert dieses Kapitel mögliche Komponenten des Systems, deren Ausfall eine Nichtverfügbarkeit eines Dienstes zur Folge hat. Diese identifizierten Komponenten stellen die SPOFs des Systems dar. Als Ursache für den Ausfall einer Komponente kommen viele Eventualitäten in Frage. Die praktische Erfahrung zeigt die Vielfalt von Ausfallursachen. Eine umfassende Liste würde den Rahmen dieser Arbeit sprengen, weshalb hier nur einige Beispiele genannt werden. Diese reichen von menschlicher Fehlbedienung und Missgeschick über Produktions- oder Softwarefehler bis hin zu Leitungsschäden durch Baggerarbeiten oder Unwetterschäden.

Durch die Komplexität und den Umfang möglicher Risiken beschränkt sich diese Arbeit auf das Rechnersystem selbst und schweift nicht in das Risikomanagement für die Planung und den Betrieb von IT-Systemen ab.

3.1.1 Infrastruktur-Anbindung

Die Energieversorgung eines Computers erfolgt im Regelfall über die folgenden Komponenten. Dabei wird von der in Deutschland üblichen 230 V Wechselspannung ausgegangen.

1. Netz des Stromanbieters

2. Hausverteilung

3. Computernetzteil

In der Regel platziert man Server-Systeme, wie das hier betrachtete System, in klimatisierten Räumen, um für eine gleichmäßige Betriebsumgebung zu sorgen. Besonders zu hohe Luftfeuchtigkeit und zu hohe Temperaturen können den Ausfall eines Systems nach sich ziehen. Deshalb stellt auch die Klimatisierung des Betriebsraums ein Ausfall-Risiko dar.

4. Klimatisierung

3.1.2 Netzwerk-Anbindung

Dieser Punkt umfasst sämtliche Verbindungen zu anderen Systemen. Im Regelfall erfolgt der Betrieb einer VoIP-Software-PBX in einem LAN auf der Basis von Ethernet und TCP/IP-Protokollen.

1. LAN, wird benötigt für die Verbindung mit SIP-Endgeräten, Clients und VoIP-Anbietern
2. Internet-Anbindung, wird benötigt für die Verbindung mit VoIP-Anbietern
3. ISDN, wird benötigt für die Verbindung mit ISDN-Endgeräte und PSTN-Anbietern
4. Analoge Leitungen, wird benötigt für die Verbindung mit analogen Endgeräten und PSTN-Anbietern
5. GSM, wird benötigt für die Verbindung mit GSM-Anbietern

3.1.3 Systemhardware

Die übliche Hardware-Plattform eines Standard-PCs besteht aus folgenden Komponenten.

1. CPU
2. Mainboard
3. RAM
4. Festplatte
5. Schnittstellenkarten für Ethernet und gegebenenfalls ISDN- oder Analog-Leitungen.
6. Lüfter, kann bei Ausfall zu einer Überhitzung bestimmter Komponenten, wie z. B. CPU führen, was wiederum zum Systemabsturz oder schlimmstenfalls zum Defekt der CPU führen kann.

3.1.4 Software

Neben dem Betriebssystem und seinen Komponenten laufen auf dem System noch weitere Prozesse, die die Dienste bereitstellen. Die folgende Liste stellt die relevanten Software-Komponenten dar.

1. Linux Kernel, Betriebssystem-Kern mit notwendigen Hardware-Treibern
2. Betriebssystem-Bibiliotheken
3. Asterisk, VoIP-Komponente
4. Apache, Web-Server
5. MySQL, Datenbank-Server
6. Postfix, E-Mail-Server

3.1.5 Risiken beim Betrieb

Schon der Betrieb eines Rechnersystems birgt gewisse Ausfallrisiken.

1. Planmäßige Wartung, Updates

2. Hacker-Angriffe

3. Fehlbedienung

3.1.6 Zusammenfassung der Ausfallrisiken

Die untenstehende Tabelle fasst die ausfall-gefährdeten Komponenten einer VoIP-Software-PBX zusammen und kategorisiert sie.

Nr.	Komponente	Kategorie
1	Netzteil	System-intern
2	Ethernet-Karte	System-intern
3	ISDN-Karte	System-intern
4	Karte für analoge Telefonleitung	System-intern
5	GSM-Karte	System-intern
6	CPU	System-intern
7	RAM	System-intern
8	Mainboard	System-intern
9	Festplatte	System-intern
10	Lüfter	System-intern
11	Betriebssystem	Software
12	Asterisk	Software
13	Apache	Software
14	MySQL	Software
15	Postfix	Software

Tabelle 4: Ausfallrisiken

3.2 Konzept der Redundanz

Aus Tabelle 4: Ausfallrisiken geht die Vielzahl von system-internen Komponenten hervor, die einen Ausfall verursachen können. Dazu kommen noch Risiken von außerhalb des Systems, wie in den Kapiteln 3.1.1 bis 3.1.5 teilweise beschrieben. Diese Vielfältigkeit an Ausfall-Szenarien macht die Forderung nach einem möglichst generischen Konzept, das möglichst viele Risiken abdeckt, deutlich.

Als allgemein gültiges Konzept bietet sich hier das Konzept der Redundanz an. Das Ziel dabei ist, für jede Komponente eine entsprechende Kopie zur Verfügung zu haben, die bei Ausfall der Hauptkomponente deren Funktion übernimmt.

3.3 Redundanz-Lösungen

Das folgende Kapitel beschreibt Konzepte, wie Redundanz bei den verschiedenen Komponenten einer VoIP-Software-PBX erreicht werden kann. Das Ziel ist der Aufbau eines HA-Clusters aus zwei Computern. Ein Computer agiert dabei als aktiver Knoten, der im Normalzustand die Dienste bereitstellt, auch Master-Knoten genannt. Der andere Computer überwacht die Funktion des Masters und übernimmt bei dessen Ausfall die entsprechenden Dienste. Dieser Knoten befindet sich also im Hot-Standby-Betrieb und wird auch als Slave-Knoten bezeichnet. Clients greifen je nach Zustand des Clusters auf die Ressourcen des Masters oder des Slaves zu. Der Zustand des Clusters ist für den Client transparent. Diese Eigenschaft der Sichtbarkeit des Clusters als ein System wird auch als „Single System Image“, kurz SSI bezeichnet. Abbildung 3: Prinzipielle Zustände eines HA-Clusters zeigt eine graphische Darstellung der beiden prinzipiellen Zustände eines HA-Clusters.

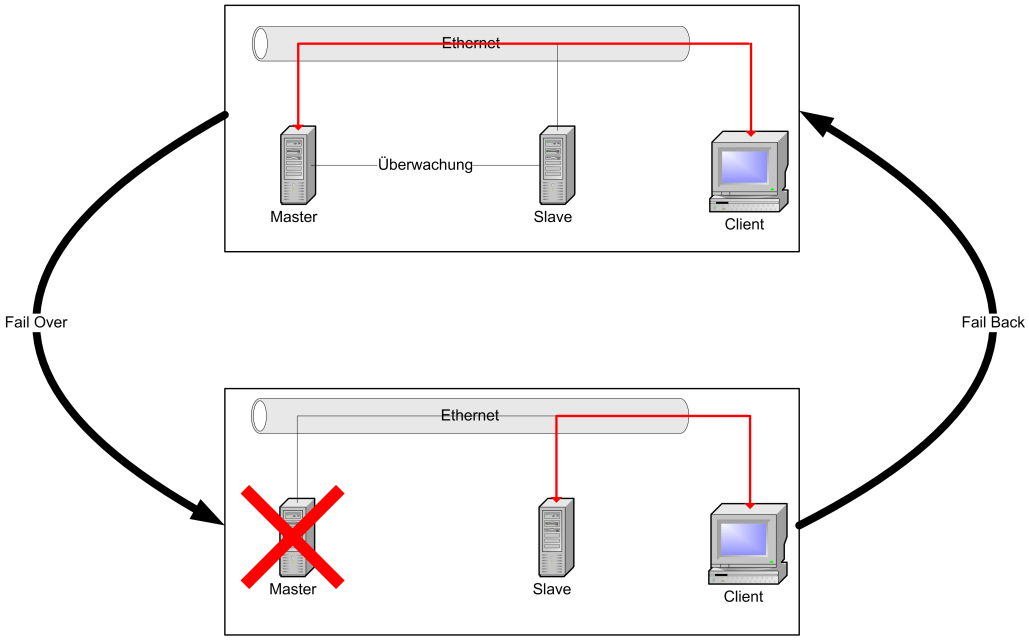

Abbildung 3: Prinzipielle Zustände eines HA-Clusters

In den Cluster-Knoten empfiehlt sich trotz der redundanten Systeme der Einsatz von RAID-Systemen, um Redundanz auf Massenspeicher-Ebene zu erreichen. Festplatten-Defekte führen nicht zwangsläufig zu einem Ausfall des Knotens, wodurch kein Fail Over erfolgt. Diese Situation kann zu inkonsistenten Daten auf dem noch aktiven Knoten führen, wenn er nicht über ein RAID-System verfügt.

3.3.1 System-externe Komponenten

Der Begriff „system-externe Komponenten“ beschreibt hierbei Komponenten, die nicht Bestandteil einer VoIP-Software-PBX sind, trotzdem aber deren Verfügbarkeit beeinträchtigen können.

Bei der Stromversorgung bietet sich neben dem Einsatz eines Geräts zur unterbrechungsfreien Stromversorgung (abgekürzt USV) der Anschluss der Knoten an getrennte Stromkreise an. Dies empfiehlt sich besonders dann, wenn die Computer über redundante Netzteile verfügen. So beeinträchtigt der Ausfall eines Stromkreises nicht den Betrieb des Clusters und löst keinen Zustandswechsel des Clusters aus.

Die Abbildung 4: Redundante Stromversorgung zeigt den Aufbau einer entsprechend redundanten Verkabelung. Die blauen Linien stellen die Verkabelung des mit zwei Stromkreisen dar. Die roten Linien kennzeichnen die redundanten Leitungen bei der Verwendung von Netzteilen und USVs mit jeweils zwei Spannungseingängen.

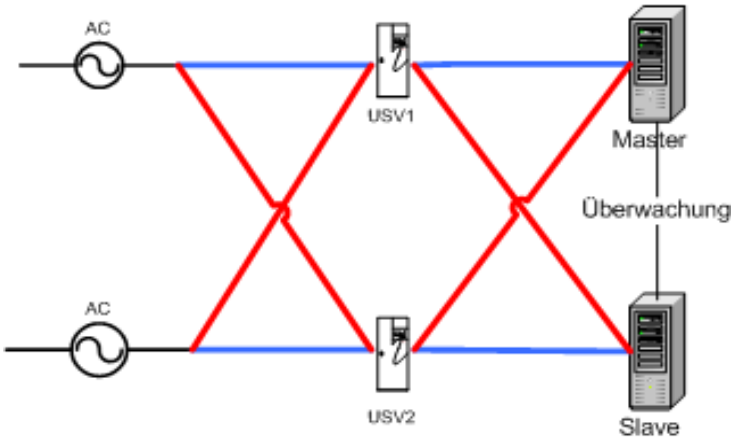

Abbildung 4: Redundante Stromversorgung

In ähnlicher Weise lassen sich redundante Ethernet-Anbindungen mit Hilfe des Spanning Tree Protokolls realisieren.

3.3.2 Fail Over Alternativen für PSTN-Leitungen

ISDN-Leitungen werden auf dem deutschen Markt in zwei Ausführungen von verschiedenen Anbietern zur Verfügung gestellt. Zum einen als S_0-Anschluss, auch Basis-Anschluß genannt. Die englische Bezeichnung lautet basic rate interface, abgekürzt BRI. Er verfügt über zwei B-Kanäle mit jeweils 64 kbit/s Bandbreite und einen D-Kanal mit 16 kbit/s Bandbreite. Basis-Anschlüsse unterteilen sich wiederum in Mehrgeräte-Anschlüsse und Anlagen-Anschlüsse. Ein Mehrgeräte-Anschluß wird hauptsächlich im Privatkundenbereich verwendet, um direkt mehrere Endgeräte mit eigenen Rufnummern (MSN) anzuschliessen. Der Anlagen-Anschluß eignet sich zum Anschluss einer ISDN-Nebenstellen-Anlage. Diese bekommt einen Rufnummernblock zugewiesen, den sie verwaltet.
Für größere Installationen eignet sich der so genannte S_2M-Anschluss, oder auch Primär-Multiplex-Anschluß genannt. Die englische Bezeichnung lautet primary rate interface, abgekürzt PRI. Er verfügt über 30 B-Kanäle mit jeweils 64 kbit/s und zwei D-Kanäle, ebenfalls mit 16 kbit/s.

3.3.2.1 Gemeinsamer S_0-Bus

Beim S_0-Anschluss handelt es sich um einen Bus. Somit können mehrere Geräte an den S_0-Anschluss hier dem NTBA (ISDN Netz-Abschluss) angeschlossen werden.

Diese Tatsache lässt sich für den Entwurf des HA-Clusters ausnutzen. Beide Knoten verfügen über eine entsprechende ISDN-Karte, die an den Bus angeschlossen wird. Dabei muss nur ausgeschlossen werden, dass der Slave-Knoten auf eingehende Rufe am ISDN reagiert.

3.3.2.2 Fail Over Switches

Verschiedene Hersteller bieten so genannte Fail Over Switches (abgekürzt FOS) an. Deren prinzipielle Funktionsweise leitet sich schon aus dem Begriff Fail Over Switch ab. Ein Fail Over Switch verfügt über mindestens einen Eingang für eine Leitung und mindestens zwei Ausgänge. Die Amtsleitung wird am Eingang in den Switch eingespeist. Master- und Slave-Knoten des HA-Clusters werden an den jeweiligen Ausgang angeschlossen. Zusätzlich verfügen einige FOS über einen Eingang für Cluster-Kommunikation oder überwachen den Status der Leitungen mit eigenen Watchdog-Prozessen.

Im Normalbetrieb schleift der FOS die Amtsleitung auf den Master-Ausgang durch. Findet nun ein Fail Over im Cluster statt, bekommt der FOS dies entweder durch die Cluster-Kommunikation oder seinen eigenen Watchdog mitgeteilt. Nun schaltet er die Amtsleitung auf den Slave-Ausgang.

Allerdings gilt es dabei zu bedenken, dass der Einsatz eines FOS wiederum SPOF darstellt. Leider war es mir im Rahmen dieser Arbeit nicht möglich, eine Teststellung für einen oder mehrere FOS zu erhalten. Ebenso lässt sich aus den Dokumentationen nicht erkennen, inwieweit die einzelnen Produkte redundant betrieben werden können.

Hersteller	Gerät	BRI	PRI	Analog	#Amt	#int	com
Beronet	bero*fos	X	X	X	4	4M+4S	-
FSV	4PFS	X	X	X	4	4M+4S	X
Wti	PLS-345	X	X	X	3	3M+3S	-
Junghanns	ISDNguard	X	X	-	4	4M+4S	X

Tabelle 5: Kurzübersicht Fail Over Switches

Eine kurze Auflistung einiger am Markt verfügbarer FOS liefert Tabelle 5: Kurzübersicht Fail Over Switches. Neben Hersteller und Gerätename enthält die Tabelle Informationen, für welche Leitungstypen der FOS geeignet ist. Die Spalte

„#Amt“ enthält die Anzahl der möglichen Amtleitung, für die der FOS verwendet werden kann. Die Spalte „#int“ gibt an wieviele interne Master- und Slave-Leitungen zur Verfügung stehen. Die Spalte „com“ gibt an, ob eine Kommunikations-Schnittstelle zum Master-Knoten vorhanden ist. Dabei bedeutet 'X', dass die Eigenschaft vorhanden ist, '-' bedeutet, dass die Eigenschaft nicht vorhanden ist.

3.3.2.3 HA-Cluster Anbindung mittels TDM over Ethernet

Eine weitere Alternative, wie Amtsleitungen an ein HA-Cluster angebunden werden können, liefert der Hersteller Redfone mit dem Produkt foneBridge2. Dieses Produkt wird in zwei Varianten angeboten, für zwei oder für vier T1/E1-Leitungen. T1 bezeichnet hierbei den amerikanischen ISDN-Standard mit 24 B-Kanälen, E1 den europäischen mit 30 B-Kanälen. TDM ist in diesem Zusammenhang die Abkürzung für „Time Division Multiplexing“, im Deutschen als Zeitmultiplexverfahren bezeichnet. Nach diesem Verfahren werden im ISDN die einzelnen B-Kanäle auf der physikalischen Drahtleitung kodiert und übertragen. Eine foneBridge2 verfügt neben den jeweiligen TDM Anschlüssen über einen Ethernet-Anschluß. Über den TDM-Anschluß wird die foneBridge2 an das PSTN angeschlossen und über den Ethernet-Anschluß an das LAN. Über diese Verbindung kann nun ein Asterisk-Prozeß mit der foneBridge2 kommunizieren und Verbindungen ins PSTN herstellen. Ebenso ist einem zweiten Asterisk-Knoten die Kommunikation mit einer foneBridge2 möglich.

Mittels der sehr flexiblen TDMoE (TDM over Ethernet)-Technologie lassen sich sehr komplexe Szenarien redundant aufbauen, da ein Asterisk-Prozeß auch mehrere TDMoE-Geräte bedienen kann. In [AMA-2007] ist der Aufbau eines HA-Clusters mit zwei Knoten und zwei foneBridge2-Geräten, die jeweils an eine eigene TDM-Leitung angeschlossen sind, beschrieben. Der dort beschriebene Aufbau weist demnach keinen SPOF auf.

Leider standen mir auch hier weder Leitungen noch Hardware zur Verfügung, um diesen Ansatz im Rahmen dieser Arbeit verifizieren zu können.

3.3.3 Redundante GSM-Anbindung

Neben der Anbindung einer VoIP-Software-PBX an das ISDN oder analoge Leitungen (englisch: Plain Old Telephone Servive, abgekürzt POTS), ist auch eine Anbindung an Mobilfunknetze nach dem GSM-Standard (Global System for Mobile Communications) möglich. Dazu benötigt die VoIP-Software-PBX eine entsprechende PCI-Steckkarte, wie sie beispielsweise von der Firma Junghanns und anderen angeboten wird. In eine solche PCI-Karte wird eine handelsübliche SIM-Karte (Subscriber Identity Module) eines Mobilfunk-Betreibers eingebaut. Die GSM-Karte stellt nun eine Verbindung zum entsprechenden Mobilfunknetz her. Diese Verbindung steht nun als Kommunikations-Kanal (channel) in der VoIP-Software-PBX zur Verfügung.
Da es sich hierbei um eine Funk-Übertragung handelt, ist das Übertragungsmedium, in diesem Fall die Luft, im Gegensatz zu 3.3.2 redundant vorhanden. Ausgehende Verbindungen sollten deshalb jederzeit von jedem Knoten aus mit seiner einen GSM-Verbindung möglich sein. Für eingehende Verbindungen muss der Mobilfunkbetreiber kontaktiert werden, da in dessen Netz die Umsetzung einer Rufnummer auf die Identifikationsnummer einer SIM-Karte erfolgt. In Quelle [ZAB-2004] wird ein entsprechendes Verfahren beschrieben. Der Mobilfunkbetreiber o2 bietet in Deutschland eine so genannte Multicard-Option an, bei der bis zu drei SIM-Karten für eine Rufnummer verwendet werden können.

3.3.4 HA-Cluster in einem IP-Netz

In diesem Kapitel werde ich verschiedene Open-Source-Projekte zur Realisierung von HA-Cluster mit Linux-Systemen vorstellen.

Das Ziel ist dabei ein Zwei-Knoten-HA-Cluster in einer Master-Slave-Konstellation zu implementieren, das die in Tabelle 4: Ausfallrisiken aufgezeigten Komponenten 1, 2 und 6 bis 15 redundant bereitstellt. Die Voraussetzung dafür sind zwei Computersysteme mit identischer Hardwarekonfiguration. Diese prototypische Demonstration bezieht sich rein auf die Bereitstellung eines SSI auf der IP-Netzwerk-Ebene. Dieser Aufbau ist damit unabhängig von den Aspekten, die in den Kapiteln 3.3.1 bis 3.3.3 dargestellt wurden und im Bedarfsfall die Komponenten 3 bis 5 abdecken. Die Punkte 1, 2 und 6 bis 10 sind durch den zweiten Rechner implizit abgedeckt. Die Punkte 11 bis 15 stellen die relevanten Netzwerk-Dienste des Systems dar, die als SSI verfügbar sein sollen. Die Realisierung dessen ist Gegenstand dieses Kapitels.

3.3.4.1 HA-Cluster mit Heartbeat

Bei Heartbeat handelt es sich um eines der wichtigsten und am weitesten verbreiteten Software-Projekte zur Realisierung von HA-Clustern aus dem Open-Source-Bereich. Es ist für annähernd jede größere Linux-Distribution verfügbar, ebenso wie für die freien BSD-Derivate FreeBSD und OpenBSD.

Auf Heartbeat, als einer der Kernkomponenten des Linux-HA-Project [ROB-2007], basiert eine Vielzahl von produktiv eingesetzten HA-Clustern, auch im kommerziellen Bereich. Dies ist ein eindeutiger Beweis für die Leistungsfähigkeit des Systems, das sich auch neben kommerziellen Produkten behaupten kann.

Der Entwurf der Software folgt den Grundsätzen von Schlankheit, Einfachheit, Robustheit und Zuverlässigkeit, wie in [ROB-2000] beschrieben. Während Heartbeat in der ersten Version nur eine sehr eingeschränkte API besaß und nur Zwei-Knoten-Cluster unterstützte, wurden diese Hürden in der Version 2 überwunden, ohne dabei die o. g. Stärken einzuschränken.

Ein HA-System benötigt grundlegend zwei Funktionalitäten. Zum einen muss der Beitritt und das Ausscheiden von Knoten aus dem Cluster festgestellt und übermittelt

werden, zum anderen muss ein Kommunikationssystem zur Übermittlung von Cluster-Nachrichten zwischen den Knoten vorhanden sein.
Einer Empfehlung von Harald Milz [MIL-1997] folgend unterstützt Heartbeat auch die Kommunikation über alternative Medien, nicht nur über IP.
Dies ist wichtig, um auch die so genannten Heartbeat-Verbindungen redundant auslegen zu können. So könnte z. B. der TCP/IP-Stack bzw. der Netzwerkkarten-Treiber in einem Knoten zu einem SPOF werden. Dies kann unter Umständen zu einem inkosistenten Cluster-Zustand führen, wenn der Slave feststellt, dass der Master keine Heartbeat-Nachrichten mehr auf der Heartbeat-Leitung sendet. In Folge dessen würde der Slave einen Fail Over durchführen. Deshalb kommen oftmals noch einfache serielle Leitungen als zusätzliche Heartbeat-Verbindungen zum Einsatz, das diese nicht vom TCP/IP-Stack des Kernel abhängen. Sollte es zum Ausfall dieses Stacks kommen, schickt der Master immer noch Heartbeat-Nachrichten auf der seriellen Leitung. Deshalb führt der Slave keinen Fail Over durch. Zwar können Clients unter Umständen nicht mehr auf das Cluster zugreifen, aber diese Einschränkung ist einer Split Brain Situation vorzuziehen. In einer solchen Situation ist der Master noch aktiv und dem zu Folge auch seine Ressourcen-Gruppen noch vorhanden. Führt nun der Slave einen Fail Over durch, weil er innerhalb des angegeben Zeitraums keine Heartbeat-Nachricht vom Master erhalten hat und ihn demzufolge als ausgefallen betrachtet, könnte dies zu einem inkonsistenten Zustand des Clusters führen. Um dies zu verhindern, besonders bei produktiv eingesetzten Clustern, wurde neben seriellen Heartbeat-Leitungen das Heartbeat-Modul STONITH entwickelt. STONITH ist die Abkürzung für den englischen Satz „Shoot The Other Node In The Head“. Hinter dieser martialischen Bezeichnung verbirgt sich der Ansatz, dass der Slave bei einem Fail Over den Master stromlos schaltet. Dies lässt sich mit fernbedienbaren Stromschaltern realisieren. So ist sichergestellt, dass eine Ressourcen-Gruppe zu einem Zeitpunkt immer nur einem Knoten zugewiesen ist.

Damit der Ausfall eines Knotens möglichst schnell festgestellt werden kann, muss das Intervall für die Heartbeat-Nachrichten entsprechend klein konfiguriert werden. Diese kleine Heartbeat-Intervalle erzeugen entsprechend mehr Datenvolumen auf den Heartbeat-Leitungen. Deshalb stellt sich die berechtigte Frage nach dem Bandbreitenbedarf einer solchen Heartbeat-Implementierung. In [ROB-2000] findet sich dazu folgende Berechnung.

$$B_{hb} = S_{hb} * 8_{bits/byte} * R_{hb} * N$$

Abbildung 5:

Bandbreitenberechnung

Heartbeat

Dabei bezeichnen:

- B_{hb} den Bandbreitenbedarf der Heartbeat-Nachrichten
- S_{hb} die durchschnittliche Größe einer Heartbeat-Nachricht, 150 byte wird hier als Durchschnittswert angegeben
- R_{hb} die Anzahl der Nachrichten pro Sekunde, hier 1
- N die Anzahl der Knoten im Cluster, hier 1000

Damit ergibt sich ein Bandbreitenbedarf von 1.200.000 bits pro Sekunde. Dies entspricht ca. 1,2% der Bandbreite eines ungeswitchten 100Mbit/s Ethernet Netzwerks. Dies zeigt, dass selbst bei einer derart großen Anzahl von Cluster-Knoten der Bandbreitenbedarf vernachlässigbar ist.

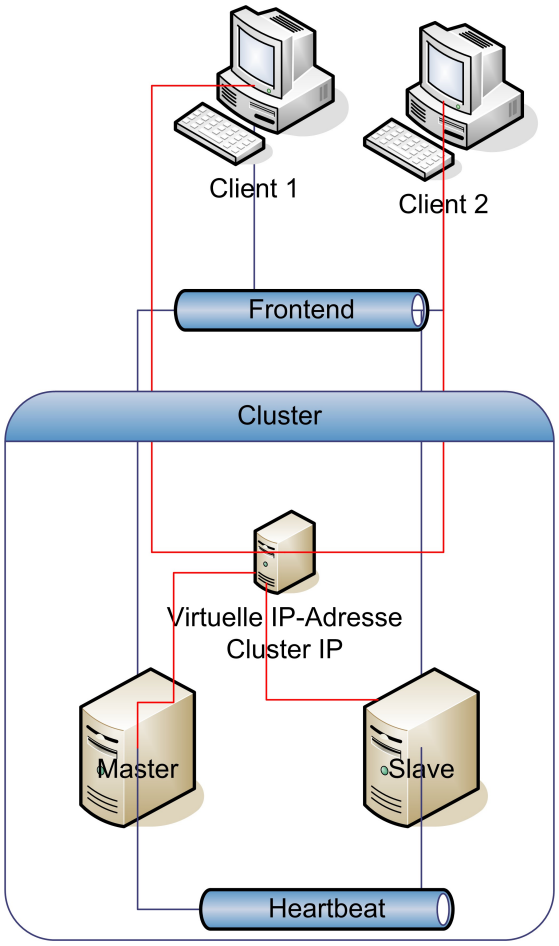

Abbildung 6: Heartbeat HA-Cluster

Die Abbildung 6: Heartbeat HA-Cluster zeigt den schematischen Aufbau des HA-Clusters mit Heartbeat. Das Cluster besteht aus dem Master- und dem Slave-Knoten und erscheint den Clients als Netzwerk-SSI durch die virtuelle IP-Adresse. Die Clients greifen über das Netz „Frontend“ auf die virtuelle IP-Adresse des Clusters zu. Je nach Zustand des Clusters reagiert Master oder Slave auf Anfragen an diese IP-Adresse. Heartbeat steuert, welcher Knoten diese Ressource besitzt. Zusätzlich verfügt das Cluster über das Netz „Heartbeat“, das ausschließlich zur Cluster-Kommunikation zwischen den Knoten dient. Voraussetzung dafür ist, dass beide Knoten über zwei Ethernet-Schnittstellen verfügen. Im Falle eines Zwei-Knoten-Clusters, wie in diesem Fall, kann dieses Netz auch mit einem einfach Ethernet-Crossover-Kabel realisiert

werden.

Die virtuelle Cluster-IP ist eine virtuelle Schnittstelle auf der physikalischen Schnittstelle des entsprechenden IP-Netzes. Linux bezeichnet Ethernet-Schnittstelle nach dem Schema eth<#>, wobei # eine Nummer beginnend mit 0 bezeichnet. Somit lautet die Bezeichnung der ersten Ethernet-Schnittstelle in einem System eth0. Die Benennung virtueller Schnittstellen folgt dem Schema eth<#1>:<#2>, wobei #1 die Nummer der physikalischen Schnittstelle bezeichnet und #2 die Nummer der virtuellen Schnittstelle auf dieser physikalischen. So bezeichnet eth0:0 die erste virtuelle Schnittstelle der ersten physikalischen Schnittstelle. Wechselt das Cluster seinen Zustand und ein anderer Knoten übernimmt die virtuelle IP-Adresse, versendet er einen so genannten „gratuitous ARP“, um allen Systemen im betreffenden Netzwerk seine MAC-Adresse als die neue MAC-Adresse zu der virtuellen IP-Adresse mitzuteilen. Dieser Mechanismus umgeht das Cachen von ARP-Anfragen auf Seiten der Clients. Durch das Cachen von ARP-Anfragen würden Clients immer noch versuchen Ethernet-Frames an die MAC-Adresse des deaktivierten Cluster-Knotens zu schicken. Ein gratuitous ARP ist eine Broadcast-ARP-Anfrage eines Hosts, bei der er seine eigene IP-Adresse (in diesem Falle die virtuelle Cluster-IP-Adresse) als Quell- und Ziel-Adresse angibt. Dieser Vorgang wird in Abbildung 7: Gratuitous ARP veranschaulicht.

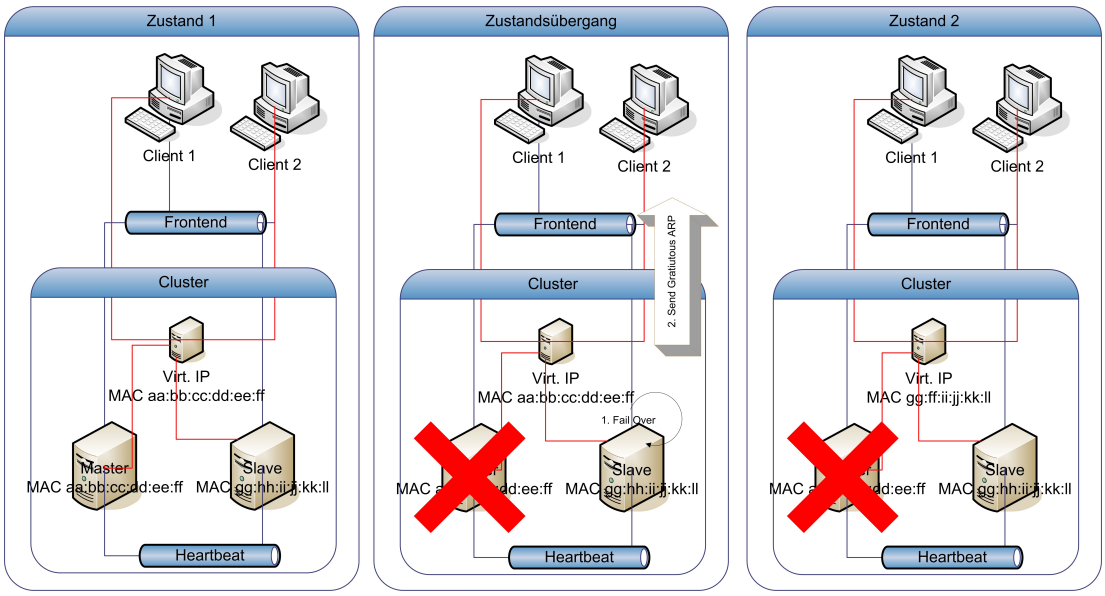

Abbildung 7: Gratuitous ARP

Auf einem Ubuntu Server lässt sich Heartbeat2 über ein Installationspaket in der Sektion „universe“ mit dem Namen heartbeat-2 installieren. Die Konfiguration von Heartbeat über drei Konfigurationsdateien im Verzeichnis /etc/ha.d/

- ha.cf

  Die Hauptkonfigurationsdatei für allgemeine Heartbeat-Parameter

- authkeys

  Authentifizierungsinformationen zur Authentifizierung der Knoten untereinander

- /var/lib/heartbeat/crm/cib.xml (bei Verwendung von Heartbeat2 und dem Cluster Ressource Manager, kurz CRM)

  Cluster Information Base CIB, Datei zur Konfiguration der Ressource-Gruppen im Cluster

- haressources (wenn der CRM nicht verwendet werden)

Bei diesen Dateien handelt es sich um ASCII-Dateien, die mit einem beliebigen Texteditor bearbeitet werden können. Die Einstellungen richten sich jeweils nach den Gegebenheiten des Netzwerks, in dem das Cluster implementiert werden soll. Das

Netzwerk für die prototypische Umsetzung wurden entsprechend Tabelle 6:

Netzwerkparameter für prototypische Umsetzung konfiguriert.

Parameter	Wert
Rechner-Name Master	cl01
Rechner-Name Slave	cl02
Frontend IP-Netz	192.168.178.0
Frontend Netzwerk-Maske	255.255.255.0
Frontend Broadcast	192.168.178.255
Frontend IP-Adresse Master	192.168.178.3
Frontend IP-Adresse Slave	192.168.178.4
Frontend Cluster-IP	192.168.178.5
Frontend IP-Adressen Clients	192.168.178.10-192.168.178.15
Heartbeat IP-Netz	10.0.0.0
Heartbeat Netzwerk-Maske	255.255.255.0
Heartbeat Broadcast	10.0.0.255
Heartbeat IP-Adresse Master	10.0.0.3
Heartbeat IP-Adresse Slave	10.0.0.4
Hearbeat Ressource-Gruppe	192.168.178.5 mysql asterisk postfix apache

Tabelle 6: Netzwerkparameter für prototypische Umsetzung

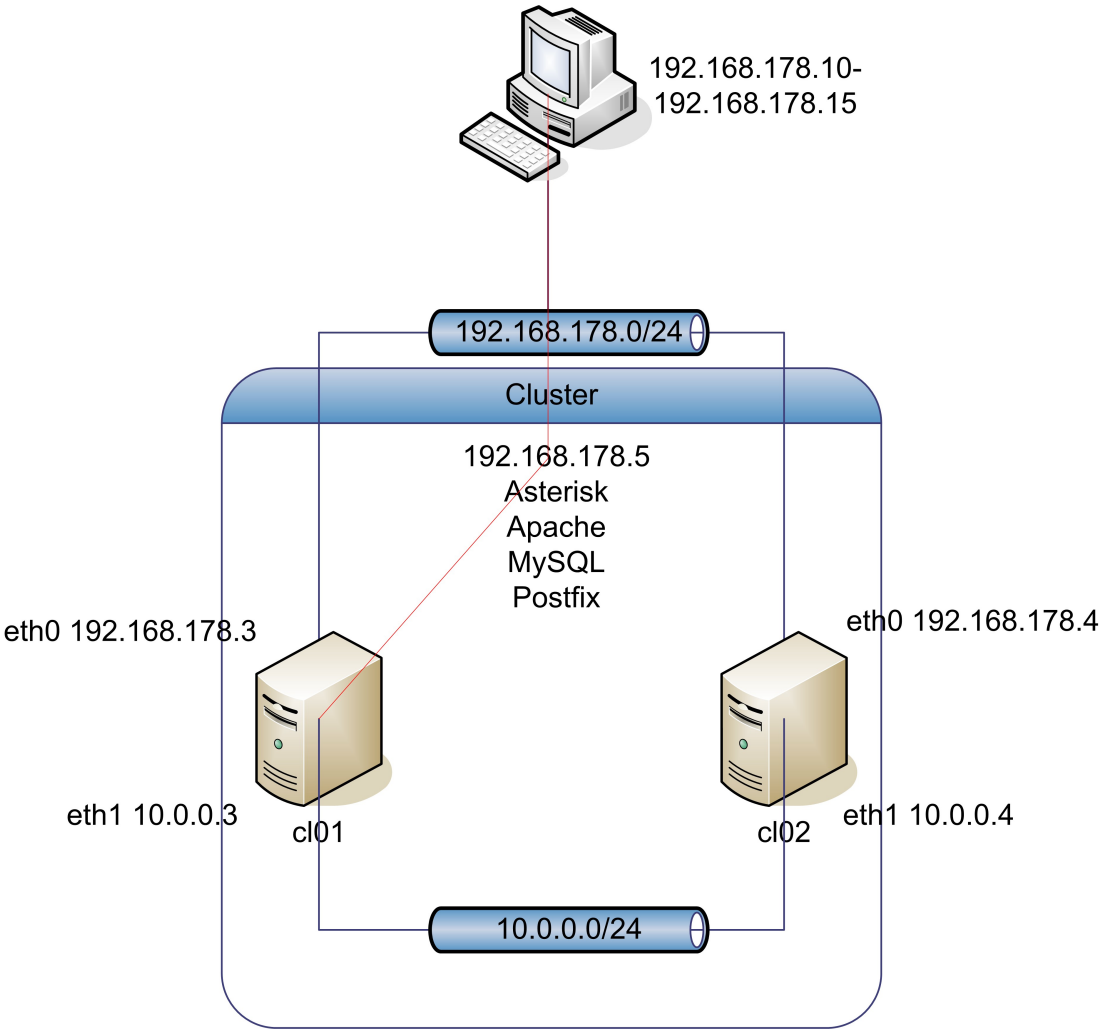

Abbildung 8: Netzwerkplan prototypische Umsetzung HA-Cluster

Aus den Daten in Tabelle 6: Netzwerkparameter für prototypische Umsetzung ergibt sich der in Abbildung 8: Netzwerkplan prototypische Umsetzung HA-Cluster gezeigte Netzwerkplan für die prototypische Umsetzung des HA-Clusters im Normalzustand. Diese Cluster-Konfiguration wird durch entsprechende Einträge in den Heartbeat-Konfigurationsdateien erreicht.

Die Datei authkeys muss auf allen Knoten identisch sein und den Zugriffsmodus 600 besitzen.

```
auth 1
1 md5 supersecretclusterkey
```

Abbildung 9: authkeys

Die Datei ha.cf muss ebenfalls auf allen Knoten identisch sein.

```
#logfacility local7
#logfile /var/log/ha-log
#debugfile /var/log/ha-debug
use_logd on
udpport 694
keepalive 1 # 1 second
deadtime 10
initdead 80
bcast eth0 eth1
#serial /dev/ttyS0 #if you use serial
#baud 19200 #if you use serial
node cl01 cl02
crm yes
auto_failback yes
```

Abbildung 10: ha.cf

In Zeilen, die ein '#'-Zeichen enthalten, wird alles nach diesem Zeichen als Kommentar interpretiert und hat keinen Einfluss auf die Konfiguration. Die Bedeutung der aktiven Direktiven und deren Parameter in dieser Datei erläutert Tabelle 7: ha.cf Direktiven.

Direktive	Parameter	Bedeutung
use_logd	on	Aktiviert die Verwendung des Heartbeat-eigenen Log-Dienstes
udpport	694	Konfiguriert UDP-Port 694 für die Cluster-Kommunikation
keepalive	1	Heartbeat-Intervall in Sekunden
deadtime	10	Zeit in Sekunden nach der ein Knoten ohne Heartbeat-Nachricht für inaktiv erklärt wird
initdead	80	Initiale Wartezeit, nur relevante nach Neustart o. ä,
bcast	eth0 eth1	Broadcast-Kommunikation über die Schnittstellen eth0 und eth1
node	cl01 cl02	Knoten des Clusters, hier cl01 und cl02
crm	yes	Aktiviert die Verwendung des Cluster Ressource Managers von Heartbeat2
auto_failback	yes	Nach Rückkehr eines ausgefallenen Knotens sollen ihm wieder seine Ressourcen zugewiesen werden

Tabelle 7: ha.cf Direktiven

Da in diesem Beispiel der CRM verwendet werden soll, muss die Konfiguration der Ressourcen über die Datei /var/lib/heartbeat/crm/cib.xml erfolgen. Da es sich hierbei um eine komplexe XML-Datei handelt, empfiehlt es sich entweder ein graphisches Konfigurationswerkzeug zu verwenden oder eine haressources-Datei zu konvertieren. In diesem Beispiel erstelle ich eine wesentlich einfachere haressources-Datei, in diesem Falle /root/haressources.temp und konvertiere diese in eine CIB. Heartbeat2 liefert ein entsprechendes Programm zur Konvertierung mit (/usr/lib/heartbeat/cts/haresources2cib.py).

```
cl01 IPAddr::192.168.178.5/24/eth0 mysql postfix
    asterisk apache2
```

Abbildung 11: /root/haressources.temp

Die Bedeutung der einzelnen Einträge in dieser Datei ist der Tabelle 8: Bedeutung haressources zu entnehmen. Jede Ressourcen-Gruppe wird in einer Zeile konfiguriert. Dabei erfolgt die Allokation der Ressourcen von links nach rechts, die Freigabe der Ressourcen von rechts nach links.

Eintrag	Bedeutung
cl01	Bevorzugter Knoten; der Knoten, dem die Ressourcen-Gruppe initial zugewiesen wird und der sie bei 'auto_failback yes' in ha.cf auch wieder erhält.
IPAddr::192.168.178.5/24/eth0	Eine virtuelle IP-Ressource Die Angabe von 'IPAddr::' ist optional, sie erhöht allerdings die Lesbarkeit. '192.168.178.5' gibt die virtuelle IP-Adresse an (siehe Abbildung 8: Netzwerkplan prototypische Umsetzung HA-Cluster). '/24' gibt die Netzwerkmaske die virtuellen IP-Adresse an. Optionale Angabe. Wenn sie nicht angegeben wird, übernimmt Heartbeat die Netzwerkmaske des entsprechenden Interfaces. 'eth0' physikalisches Interface, optional. Wenn es nicht angegeben wird, ordnet Heartbeat anhand der Routing-Einträge die virtuelle IP-Adresse dem entsprechenden Interface zu.
mysql postfix asterisk apache2	Start-/Stop Skripte Findet Heartbeat in einem der Verzeichnisse /etc/init.d oder /etc/ha.d/resource.d ein Skript mit einem entsprechenden Namen, so führt es dieses

Eintrag	Bedeutung
	mit den Parametern 'start' bzw. 'stop' aus (siehe unten).

Tabelle 8: Bedeutung haressources

Die Konvertierung erfolgt über das Kommando

```
cladmin@cl01~$ sudo python
    /usr/lib/heartbeat/cts/haresources2cib.py
    /root/haresources.temp >
    /var/lib/heartbeat/crm/cib.xml
```

'cladmin@cl01~$ ' kennzeichnet hierbei die Shell des Benutzers cladmin auf dem Knoten cl01. Das Benutzerkonto cladmin wurde bei der Installation des Systems angelegt, analog hierzu ist auf dem Knoten cl02 zu verfahren.

Auch diese Datei muss auf beiden Knoten identisch sein.

Da Heartbeat als Cluster Manager das Starten der Dienste auf den Knoten verwaltet, muss der Start von Heartbeat beim Systemstart sichergestellt werden. Dies geschieht über das Kommando

```
cladmin@cl01~$ sudo update-rc.d heartbeat defaults
```

Dieser Befehl muss auf beiden Knoten ausgeführt werden. Gleichzeitig dürfen die von Heartbeat gesteuerten Dienste **nicht** beim Systemstart automatisch gestartet werden. Um dies sicherzustellen muss auf beiden Knoten folgende Sequenz von Kommandos ausgeführt werden.

```
cladmin@cl01~$ sudo update-rc.d -f postfix remove
cladmin@cl01~$ sudo update-rc.d -f apache2 remove
cladmin@cl01~$ sudo update-rc.d -f asterisk remove
cladmin@cl01~$ sudo update-rc.d -f mysql remove
```

Die von Heartbeat gesteuerten Dienste müssen vor dem Start von Heartbeat auf beiden Knoten gestoppt werden. Dies wird mittels folgender Kommando-Sequenz erreicht.

```
cladmin@cl01~$ sudo /etc/init.d/postfix stop
cladmin@cl01~$ sudo /etc/init.d/asterisk stop
cladmin@cl01~$ sudo /etc/init.d/mysql stop
cladmin@cl01~$ sudo /etc/init.d/apache2 stop
```

Nun kann auf beiden Knoten Heartbeat gestartet werden.

```
#sudo /etc/init.d/heartbeat start
```

Die Konfiguration der Dienste Apache, MySQL und Postfix ist an dieser Stelle nicht von Bedeutung. Asterisk wird entsprechend [ACK-2007] für zwei dynamische SIP-Clients und mit einem einfachen Dialplan, der eine Nebenstellen-Nummer (Extension) pro SIP-Client enthält, konfiguriert.

4 Shared Storage im VoIP-HA-Cluster

4.1 Einschränkungen des bisherigen Clusters

Der in Abschnitt 3.3.4.1 vorgestellte Prototyp für ein VoIP-HA-Cluster deckt alle geforderten Redundanz-Eigenschaften ab. Allerdings ist seine praktische Verwendbarkeit noch sehr eingeschränkt. Diese Einschränkungen werden deutlich, wenn man die spätere Verwendung des Gesamtsystems im Betrieb betrachtet. Dabei werden auf dem aktiven System Daten erzeugt und auf der Festplatte gespeichert bzw. Dateien werden verändert. Auch im Datenbank-System wird es zu Änderungen der Daten kommen, die wiederum persistent auf der Festplatte gespeichert werden. In der jetzigen Konstellation des Clusters käme es somit zu einem inkosistenten Datenbestand im Cluster, da die Änderungen der Daten auf dem Master nicht auf den Slave repliziert werden.

4.2 Datenreplikationsalternativen im HA-Cluster

Zur Vermeidung dieser Inkonsistenzen muss eine der folgenden Alternativen implementiert werden.

- Manuelle Replikationslösung
- NAS-System
- Hardware-System
- DRDB

4.2.1 Manuelle Replikationslösung

Dieser Lösungsansatz implementiert eine Menge von Skripten bzw. Programmen, die in zeitgesteuerten Intervallen, die relevanten Daten vom Master auf den Slave

replizieren. Zusätzlich müssen die Daten vor einem Fail Back des Masters vom Slave auf den Master synchronisiert werden.

Dieser Ansatz weist einige konzeptionelle Schwachstellen auf, die ihn für eine Realisierung ausschließen. Findet während eines solchen zeitgesteuerterten Synchronisationslaufs ein Fail Over statt, befindet sich der Slave in einem inkosistenten Zustand und der Master ist nicht verfügbar. Damit ist das Cluster in einem inkonsistenten Zustand, der nicht akzeptabel ist. Der Zeitaufwand zur Implementierung eines entsprechenden Transaktion-Mechanismus wäre im Verhältnis zu anderen Alternativen sehr hoch. Findet der Fail Over zwischen zwei Synchronisationsläufen statt, ist der Slave zwar nicht auf dem aktuellen Stand, aber noch konsistent.

Eine weitere Hürde bei der Realisierung dieses Ansatzes wäre das Datenbank-System. Denkbar wäre der Export der Daten auf dem Master, ein Kopieren der Daten auf den Slave und ein Import der Daten dort. Ein solcher Vorgang ist gerade bei großen Datenmengen sehr zeitintensiv. Damit steigt die Wahrscheinlichkeit, dass während einer solchen Replikation ein Fail Over stattfindet. Deshalb müsste ein Mechanismus implemetiert werden, der ein Starten der Datenbank durch Heartbeat verhindert, während Daten importiert werden. Alternativ könnte der MySQL-eigene Replikationsmechanismus für die Replikation der Daten auf den Slave verwendet werden. Allerdings muss dazu ständig eine Instanz von MySQL auch auf dem Slave laufen, was bedeutet, dass MySQL aus der Ressourcen-Gruppe von Heartbeat entfernt werden muss.

4.2.2 NAS-System

Wie oben dargelegt scheidet eine zeitgesteuerte Synchronisation als Replikationslösung aus. Sobald Daten vom Master auf ein Festplattensystem geschrieben werden, müssen sie auch dem Slave bereit stehen. Dazu bietet sich die Speicherung der Daten im Netzwerk an. Die einfachste Lösung hierfür stellt ein so

genannter Netwock Attached Storage (abgekürzt NAS) dar. Hierbei handelt es sich um ein System, das u. a. in einem IP-Netzwerk Festplattenspeicher über Protokolle wie SMB (Server Message Block) für Microsoft Windows Dateifreigaben oder NFS (Network Fileystem) für Linux/Unix[1]-Systeme bereitstellt.

Gegen eine solche Lösung spricht, dass ein NAS-System als zusätzlicher Kostenfaktor zu Buche schlagen würde. Viel schwerer wiegt allerdings die Tatsache, dass es einen SPOF im Cluster darstellen würde.

4.2.3 Hardware-Lösungen für Shared Storage

Bei vielen Cluster-Lösungen für kommerziell genutzte Plattformen kommen Hardware-Lösungen zur Bereitstellung von Shared Storage zum Einsatz. Der Begriff Shared Storage bezeichnet hierbei einen Festspeicher, den mehrere Hosts nutzen. Zur Anbindung an die Hosts verfügen derartige Storage-Systeme meist über Fiber-Channel- oder multihost-SCSI-Schnittstellen. Diese erfordern entsprechende Controller in den Hosts. Solche Systeme sind für die Verwendung in Hochverfügbarkeits-Umgebungen ausgelegt und weisen eine entsprechende Zuverlässigkeit auf. Für die meisten Anwendungsfälle einer VoIP-Software-PBX dürften sie allerdings außerhalb des finanziellen Rahmens liegen.

4.2.4 Datenreplikation zwischen Cluster-Knoten mit DRBD

Daraus leitet sich die Forderung nach einer vergleichsweise kostengünstigen, zuverlässigen, flexiblen und universellen Lösung zur Datenreplikation zwischen zwei Cluster-Knoten ab. Philipp Reisner beschreibt in seiner Arbeit [REI-2000] über DRBD (Distributed Replicated Block Device) ein Verfahren, das auch als Netzwerk-RAID1 beschrieben werden kann.

1 UNIX is a registered trademark of The Open Group

4.2.4.1 Funktionsweise von DRDB

Der Ansatz von DRBD folgt dabei der Idee, die relevanten Daten im Cluster auf jedem Knoten vor zuhalten. Jede Änderung im Datenbestand löst eine Übertragung der Änderungen an alle Knoten aus. Diese Übertragung von Veränderungen erfolgt über ein normales IP-Netzwerk. Bestenfalls fängt man dazu die Daten ab, bevor sie auf die lokale Festplatte geschrieben werden und überträgt sie mit einem gesicherten Transaktionsmechanismus auf die anderen Knoten.

In einem modernen Betriebssystem erfolgen sämtliche Zugriffe auf die Hardware ausschließlich über das Betriebssystem [TAN-1995]. Im Falle von Linux übernimmt der Linux-Kernel diese Aufgabe. Aus diesem Grund ist der Linux-Kernel auch der Ansatzpunkt von DRBD. Dies wird deutlich wenn man sich den Datenfluss innerhalb des Kernels betrachtet, wie in Abbildung 12: Datenfluss in einem einfach System [REI-2000] dargestellt.

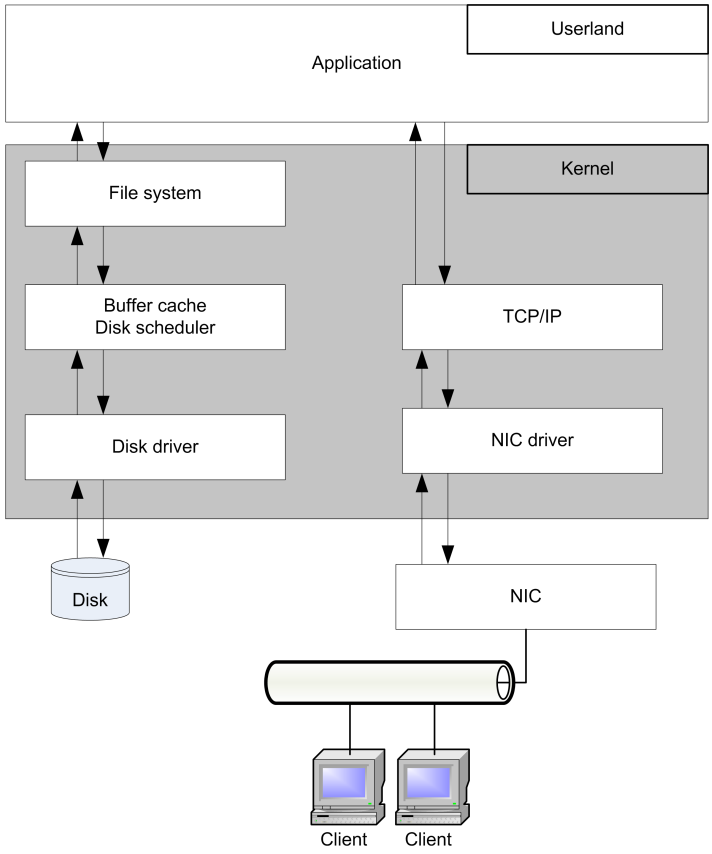

Abbildung 12: Datenfluss in einem einfach System

Die schematische Einbindung von DRBD in den Linux-Kernel zeigt die Abbildung 13: Datenfluss DRDB im Kernel nach [REI-2000].

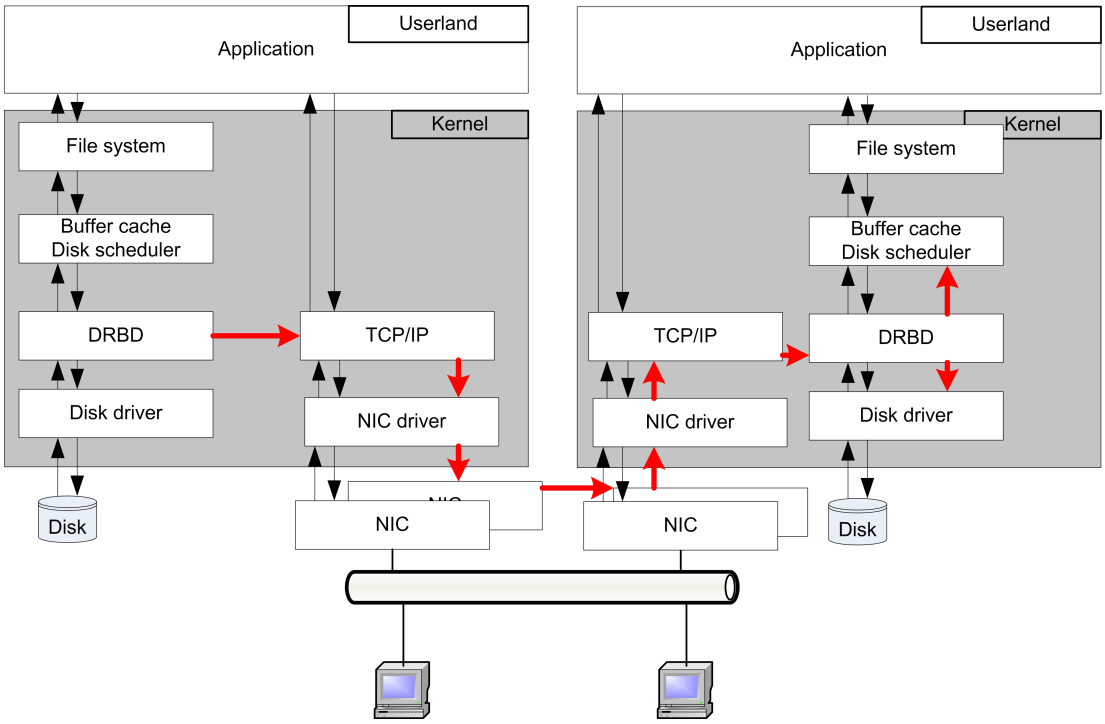

Abbildung 13: Datenfluss DRDB im Kernel

Ein DRBD-Gerät kennt zwei Zustände, die sich jederzeit ändern können:

- Primary

  Das DRBD-Gerät kann zum Schreiben und Lesen geöffnet werden. Es kann mit DRBD-Geräten auf anderen Knoten, die im sekundären Zustand sind, verbunden werden.

- Secondary

  Das DRBD-Gerät kann nur zum Lesen geöffnet werden. Es kann mit anderen DRBD-Geräten auf anderen Knoten verbunden sein.

[REI-2000]

Die in [REI-2000] behandelte frühe Version von DRBD ist für die Verwendung in HA-Clustern ausgelegt und ist nicht für die Verwendung mit einem Shared-Disk Filesystem ausgelegt. D. h. in einem Cluster kann sich nur ein DRBD-Gerät im primären Zustand

befinden und Lese-/Schreib-Operationen durchführen. Die übrigen DRBD-Geräte befinden sich im sekundären Zustand.
DRBD verfügt über drei Protokolle zur Übertragung der Daten von einem DRBD-Gerät im primären Zustand zu DRBD-Geräten im sekundären Zustand. [REI-2000] nennt diese Protokolle „Protokoll A“, „Protokoll B“ und „Protokoll C“. Sie unterscheiden sich in der Ausführungsgeschwindigkeit und Transaktionssicherheit. Dies ist nötig, um die Daten-Konsistenz sicherzustellen, auch in IP-Netzwerken, in denen die Übertragungsgeschwindigkeit nicht garantiert ist. Protokoll A weist dabei die höchste Geschwindigkeit und die geringste Sicherheit der drei Protokolle auf. Im Gegensatz dazu garantiert Protokoll C die Transaktionssicherheit, dafür werden allerdings Einbußen bei der Geschwindigkeit in Kauf genommen. Protokoll B ist schneller als Protokoll C, allerdings schließt es Split Brain Probleme nicht vollständig aus.
Nach einem Zustandswechsel führt DRBD automatisch Synchronisationsoperationen durch, um die Konsistenz der Daten sicherzustellen.

4.2.4.2 DRDB im VoIP-HA-Cluster

Da bei der prototypischen Umsetzung für diese Arbeit die I/O-Geschwindigkeit nicht von Bedeutung ist, sondern der Schwerpunkt auf der Integrität des Datenbestands liegt, kommt bei der praktischen Umsetzung Protokoll C zum Einsatz.

Der folgende Abschnitt beschreibt die Erweiterung des bestehenden HA-Clusters um ein DRBD-basiertes Shared-Storage-Gerät, das Daten enthält, die im Falle eines Fail Over jederzeit dem Client konsistent zur Verfügung stehen müssen. Dazu wird auf jedem Knoten eine zusätzliche Partition auf der lokalen Festplatte erzeugt. Im nächsten Schritt erfolgt die Konfiguration dieser Partitionen zur Verwendung als DRBD-Gerät und die Erzeugung eines Dateisystems auf diesem. Anschließend werden beide Systeme dahingehend umkonfiguriert, dass relevante Daten auf dem DRBD-Gerät

gespeichert werden. Im letzten Schritt übernimmt Heartbeat durch hinzufügen des entsprechenden DRDB-Geräts und Dateisystems zur Ressourcen-Gruppen die Steuerung der beiden Komponenten. Im Normalzustand des Clusters befindet sich das DRBD-Gerät des Masters im primären Zustand, das des Slaves im sekundären Zustand. Alle Änderungen der Daten auf dem DRBD-Gerät des Masters werden zum Slave übertragen. Kommt es zum Fail Over, wechselt das DRBD-Gerät des Slaves in den primären Zustand und stellt die Daten bereit.

Zum Einsatz von DRBD auf einem Ubuntu Server stehen die entsprechenden Pakete drbd0.7-module-source, drbd0.7-utils und drbdlinks in der Sektion „universe“ zur Verfügung. Das Paket drbd0.7-module-source enthält den Quellcode für das DRBD-Kernelmodul in der Version 0.7. Aus diesem Paket muss vorab ein Modul für den verwendeten Kernel übersetzt werden. Dies geschieht mit dem Programm module-assistant, das im gleichnamigen Paket zur Verfügung steht. Die dafür benötigte Entwicklungsumgebung wird über die Pakete build-essential, linux-headers-2.6.15-29-server und linux-headers-2.6.15-29 installiert. Mit dem Kommando

```
cladmin@cl01~$ sudo m-a a-i drbd0.7
```

wird das Kernelmodul gebaut und installiert. Nach Abschluss der Installation kann es mit dem Kommando

```
cladmin@cl01~$ sudo modprobe drbd
```

in den Kernel geladen werden.
Auf dem Knoten cl02 ist die Installation analog durchzuführen.

Auf beiden Systemen wird eine Partition /dev/hda3 einegrichtet, die als DRBD-Gerät verwendet wird. Die Konfiguration von DRBD erfolgt über die Datei /etc/drbd.conf, die auf beiden Knoten identisch ist. Die DRBD-Kommunikation erfolgt in diesem Beispiel über das Heartbeat-Netz.

```
resource r0 {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' |
wall ; sleep 60 ; halt -f";
 startup {
    degr-wfc-timeout 120;    # 2 minutes.
  }
 disk {
    on-io-error   detach;
 }
 net {
  }
 syncer {
    rate 10M;
    group 1;
    al-extents 257;
  }
 on cl01 {
    device     /dev/drbd0;
    disk       /dev/hda3;
    address    10.0.0.3:7788;
    meta-disk  internal;
  }
 on cl02 {
    device    /dev/drbd0;
    disk      /dev/hda3;
    address   10.0.0.4:7788;
    meta-disk internal;
  }
}
```

Abbildung 14: HA-Cluster DRBD-Konfiguration

Durch das Kommando

```
cladmin@cl01~$ sudo drbdadm up all
```

wird das DRBD-Gerät /dev/drbd0 initialisiert. Dieser Befehl muss auf beiden Knoten ausgeführt werden.

Auf dem ersten Knoten wird das DRBD-Gerät in den primären Zustand versetzt, um ein Dateisystem darauf anlegen zu können. Dies erfolgt mittels des Kommandos

```
cladmin@cl01~$ sudo drbdadm -- --do-what-I-say primary all
```

Die Einrichtung eines XFS[2]-Dateisystems auf /dev/drbd0 erfolgt über das Kommando

```
cladmin@cl01~$ sudo mkfs.xfs /dev/drbd0
```

Auf beiden Knoten erfolgt das Anlegen des Mount-Punktes /mnt/share für das DRBD-Gerät über den Befehl

```
cladmin@cl01~$ sudo mkdir /mnt/share
```

Das Programm drbdlinks steuert, welche Pfade aus dem bestehenden Dateisystem auf das DRBD-Gerät verlinkt werden sollen. Es wird über die Datei /etc/drbdlinks.conf konfiguriert.

```
mountpoint('/mnt/share')
link('/etc/test1')
link('/etc/test2')
```

Abbildung 15: drbdlinks.conf Beispiel

In dieser Datei wird zuerst ein Mount-Punkt angegeben, in diesem Fall /mnt/share und danach ein oder mehrere Ziele, die in den Mount-Punkt verlinkt werden sollen. Das in Abbildung 15: drbdlinks.conf Beispiel gezeigte Beispiel erzeugt zwei Links /etc/test1 und /etc/test2 auf die Pfade /mnt/share/etc/test1 und /mnt/share/etc/test2. Beim Start vom drbdlinks verschiebt es die originalen Verzeichnisse /etc/test1 und /etc/test2 und ersetzt sie durch o. g. Links auf das DRBD-Gerät. Beim Stoppen entfernt es die Links und stellt die originalen Verzeichnisse wieder her.

2 XFS: http://oss.sgi.com/projects/xfs/

Diese Konfiguration wird nun zur Heartbeat-Ressourcengruppe hinzugefügt.

```
cl01 drbddisk::r0
    Filesystem::/dev/drbd0::/mnt/share::xfs drbdlinks
    IPAddr::192.168.178.5/24/eth0 mysql postfix
    asterisk apache2
```

Abbildung 16: /root/haressources.temp mit DRBD-Ressource

Damit verfügt nun jeweils der aktive Knoten über das Dateisystem /mnt/share. Über die drbdlinks-Ressource werden beliebige Pfade auf das DRBD-Gerät verlinkt.
In dieser Konstellation stellt das HA-Cluster eine Lösung zur Ausfallsicherheit dar, die alle gestellten Forderungen in dieser Hinsicht abdeckt.

5 Lastverteilungskonzept für ein VoIP-Cluster

Mit dem in Abschnitt 4.2.4 vorgestellten System steht nun ein HA-Cluster für die VoIP-PBX zur Verfügung. Dabei kommt im Normalzustand dem Slave keine weitere Aufgabe zu, als die Heartbeat-Nachrichten des Masters zu überwachen. Aus diesem drängt sich die berechtigte Frage auf, ob es nicht möglich wäre, den Slave Knoten für weitere Aufgaben zu nutzen und so den Master zu entlasten.

5.1 Lastverhalten einer VoIP-PBX

Bei der Betrachtung der genutzten Dienste auf der VoIP-PBX fällt auf, dass sich die Systemlast proportional zur Anzahl der Verbindungen über die VoIP-PBX verfällt. Dabei schlagen insbesondere Verbindungen zu Buche, bei denen die Endgeräte unterschiedliche Codecs verwenden. In diesem Fall muss die PBX die Sprachdaten zwischen den beiden Codecs konvertieren. Diese Konvertierungen stellen besonders rechenintensive Aufgaben dar und belasten das System entsprechend.
Die Last der restlichen Dienst auf dem System ist marginal, da sie nur zu Administrationszwecken dienen und dies keine kontinuierliche Last erzeugt.

5.2 Lastverteilung durch Load Balancing auf Netzwerk-Ebene

Um dem Konzept des SSI weiter zu folgen, muss das bisherige HA-Cluster um eine Funktion zur Lastverteilung erweitert werden. Der Aufbau eines HPC- oder Grid-Clusters scheiden dabei aus konzeptionellen Gründen aus. Da diese beiden Cluster-Typen eigene Programmiermodelle voraussetzen, können die vorhandenen Applikationen darauf nicht angewendet werden. Da die Vorgabe allerdings lautet, die bestehenden Applikationen weiter zu verwenden, scheiden diese beiden Alternativen von vornherein als Lösungsansätze aus.

Dabei bleibt als einzige Alternative der Aufbau eines Load-Balancing-Clusters. Dabei muss die Lastverteilung auf Netzwerk-Ebene erfolgen und es darf kein dedizierter Load-Balancer eingesetzt werden. Dieser würde einen SPOF darstellen und die bisher erreicht Verfügbarkeit des Clusters reduzieren. Deshalb muss die Lastverteilungs-Komponente auf den Cluster-Knoten selbst platziert werden.

5.2.1 Load-Balancer mit ipvs[3] und ldirectord

Die für die Realisierung einer solchen Lösung benötigten Komponenten sind als Open-Source-Software auch für den Ubuntu Server verfügbar. Dabei handelt es sich zum einen um eine Reihe von Kernel-Modulen des Linux Virtual Server Projekts [ZHA-1999], die unter dem Namen ipvs geführt werden und zum anderen um ein Programm namens ldirectord (http://www.vergenet.net/linux/ldirectord/) von Simon Horman (http://www.vergenet.net/~horms/about_me.shtml).

Ipvs operiert als Switch auf OSI-Layer 4 für TCP- und UDP-Protokolle.

Ursprünglich war ipvs darauf ausgelegt, dedizierte Load-Balancer zu realisieren. Ein solcher Aufbau besteht aus mindestens einem Load-Balancer und mindestens zwei realen Servern, die die eigentlichen Dienste bereitstellen. Dabei stellt ein Client eine Anfrage an eine IP-Adresse des Load-Balancers. Dieser leitet nach verschiedenen Algorithmen die Anfrage an einen der realen Server weiter, der sie abarbeitet. Nach der Abarbeitung wird das Ergebnis an den Client zurück geschickt. Hinter einer Cluster-IP-Adresse des Load-Balancers verbirgt dieser eine Vielzahl realer Server, die somit für die Clients nicht sichtbar sind. Dies entspricht dem Konzept des SSI.

In einer so genannten *streamline*-Konfiguration verteilt der Load-Balancer nicht nur Client-Anfragen an andere reale Server, sondern er agiert ebenfalls als realer Server. D. h. er leitet auch Anfragen an lokale Prozesse weiter.

Das Programm ldirectord dient zur Steuerung des Load-Balancers. Über seine Konfiguration wird pro Dienst ein Pool von realen Servern definiert. Zusätzlich

3 ipvs: ip virtual server

überwacht ldirectord die Verfügbarkeit der realen Server. Fällt ein realer Server aus, wird er dynamisch aus dem Pool entfernt und es werden keine Client-Anfragen mehr an diesen realen Server verteilt.
Somit lässt sich durch konsequente und gradlinige Umsetzung dieses Konzept folgendes System skizzieren.

- Beim Systemstart starten beide Knoten alle relevanten Dienste (Apache, MySQL, Postfix, Asterisk), sie werden nicht mehr von Heartbeat verwaltet.
- Nur ldirectord steht unter der Verwaltung von Heartbeat.
    - ldirectord befindet sich auf dem Master-Knoten in einer streamline-Konfiguration mit Master- und Slave-Knoten als reale Server für alle Dienste. Fällt auf einem Knoten einer der Dienste aus, wird er aus der Lastverteilung genommen. Ist der Dienst wieder verfügbar, wird er dynamisch wieder in der Lastverteilung aufgenommen.
    - Auf dem Slave-Knoten befindet sich ldirectord in einer streamline-Konfiguration, die nur den Slave-Knoten als realen Server beinhaltet.
    - Im Normalzustand des Clusters ist ldirectord auf dem Master aktiv.
    - Fällt der Master aus, führt Heartbeat einen Fail Over durch und startet ldirectord auf dem Slave.
    - Ist der Master wieder verfügbar, wird die Ressource ldirectord wieder ihm zugewiesen.

Vor der Umsetzung eines solchen Systems besteht die Notwendigkeit einige Problematiken dabei zu betrachten und Lösungsansätze dafür zu entwickeln. In den folgenden Abschnitten werden diese aufgezeigt und diskutiert.

5.3 Dynamische Nebenstellen-Suche

In der skizzierten Konstellation mit aktiven Asterisk-Instanzen auf zwei unterschiedlichen Rechnern entscheidet der Load-Balancer, auf welchen realen

Asterisk-Server er eine SIP-Registrierungsanfrage eines Clients weiterleitet. Aufgrund dieser dynamischen Verteilung ist nicht vorhersehbar, auf welchem Knoten welcher Client registriert ist. In einem Szenario, bei dem Client A auf Server X registriert ist und Client B auf Server Y, würde ein Rufaufbau von A nach B oder umgekehrt fehl schlagen. Dies liegt in der Tatsache begründet, dass Server X keine Kenntnis von den registrierten Nebenstellen auf Server Y hat und vice versa.

Prinzipiell sind zwei Ansätze zur Lösung dieser Problematik denkbar, zum einen eine zentrale Registrierungsstelle und zum anderen ein dynamischer, dezentraler Ansatz. Der zentrale Ansatz würde in diesem Fall eine weitere Komponente im System bedeuten, somit einen SPOF für den ein Redundanz-Konzept entwickelt werden muss. Weiterhin würde sich dabei wieder die Frage der Lastverteilung stellen.

Im Gegensatz dazu beschreibt Marc Spencer in [SPE-2004] ein dezentrales Konzept zur Auffindung von Rufnummern namens DUNDi. Dabei handelt es sich um eine Asterisk-Komponente, die in den Dialplan eingebunden werden kann. Einem Asterisk-Server wird dabei in seiner Konfiguration eine Menge von Servern, so genannte Peers, da es sich um einen Peer-To-Peer-Ansatz handelt, eingetragen. Der logische Ablauf zur Auffindung einer Rufnummer im Dialplan umfasst eine Verzweigung. Im ersten Schritt schaut ein Asterisk-Server, ob die gesuchte Nebenstelle bei ihm registriert ist. Trifft dies zu, baut er die Verbindung auf. Trifft dies nicht zu, erfolgt eine Anfrage nach dieser Nummer bei allen Peers. Ist die Nummer bei einem Peer registriert, schickt dieser alle notwendigen Informationen an den anfragenden Server. Ist die Nummer bei keinem Peer registriert, schlägt der Rufaufbau fehl.

Eine beispielhafte Implementierung einer DUNDi Infrastruktur auf Basis von Asterisk wird in [RIC-2006] gegeben.

5.4 Ein gemeinsames Dateisystem im Cluster

Die in Kapitel 4 aufgestellte Forderung nach einer konsistenten Datenhaltung trifft auch für den hier vorgestellten Cluster-Typ. Eine konsequente Umsetzung dieses Ansatzes fordert sogar mehr. Bei der in Kapitel 4 aufgezeigten Lösung befindet sich zu jeder Zeit nur ein Knoten im aktiven Zustand, d. h. nur ein Knoten greift auf den Shared Storage zu.
Nun sollen aber beide Knoten aktiv sein. Diese Tatsache zieht die im folgenden behandelten Problematiken nach sich.

5.4.1 Neue DRBD-Semantiken

In der bisher verwendeten Version unterstützt DRBD nur ein Gerät im primären Zustand. Damit ist es unmöglichen von beiden Knoten aus schreibend auf ein DRBD-Gerät zuzugreifen. In der aktuellen Version 8.2.0 soll DRBD mehrere Geräte im primären Zustand unterstützen. Lars Ellenberg beschreibt in [ELL-2007] wie diese Erweiterung in DRBD umgesetzt wird. Diese Eigenschaften ist Gegenstand der aktuellen Entwicklungen von DRBD und kann noch nicht als stabil und ausgereift betrachtet werden. Da diese Version nicht als vorkompiliertes Paket vorliegt, muss die Software auf dem Zielsystem übersetzt werden.

```
resource r0 {
  protocol C;
  incon-degr-cmd "echo '!DRBD! pri on incon-degr' |
wall ; sleep 60 ; halt -f";
 startup {
    degr-wfc-timeout 120;    # 2 minutes.
  }
 disk {
    on-io-error   detach;
 }
 net {
     allow-two-primaries;
  }
 syncer {
    rate 10M;
    group 1;
    al-extents 257;
  }
 on cl01 {
    device     /dev/drbd0;
    disk       /dev/hda3;
    address    10.0.0.3:7788;
    meta-disk  internal;
  }
 on cl02 {
    device    /dev/drbd0;
    disk      /dev/hda3;
    address   10.0.0.4:7788;
    meta-disk internal;
  }
}
```

Abbildung 17: drbd.conf für Version 8

Mit dem Eintrag

```
allow-two-primaries;
```

in der Konfigurationsdatei auf beiden Knoten kann nun das entsprechende Gerät auf beiden Knoten in den primären Zustand versetzt werden.

5.4.2 Open-Source Cluster Dateisysteme

Zwar verfügt das Cluster nun über ein Shared Storage Gerät, das von beiden Knoten beschrieben werden, allerdings sind herkömmliche Linux-Dateisysteme nicht für die simultane Einbindung auf mehreren Rechnern geeignet.
Dazu werden spezielle Cluster-Dateisysteme benötigt, die über entsprechende Locking Mechanismen verfügen, um auch bei simultanem Zugriff die Datenkonsistenz gewährleisten zu können. Die Recherche nach entsprechenden Dateisystemen, die sich für die Verwendung eignen, brachte zwei Ergebnisse hervor, OCFS2[4] und GFS[5]. Eine vergleichende Übersicht zwischen beiden Dateisystemen liefert [SEI-2007].
Aufgrund der dort erwähnten einfacheren Handhabbarkeit von OCFS2 fiel die Wahl auf dieses Cluster-Dateisystem zur Implementierung eines Prototyps.

5.4.2.1 Implementierungsversuch von OCFS2 mit direktem Zugriff auf ein DRBD-Gerät

Die Dateisystem-Treiber von OCFS2 sind im Ubuntu Server enthalten, das Paket ocfs2-tools beinhaltet die benötigten Programme zur Verwaltung des Dateisystems.
Auf beiden Systemen muss die Datei /etc/ocfs2/cluster.conf vorhanden sein.

4 Oracle Cluster File System 2

5 Global File System

```
node:
        ip_port = 7777
        ip_address = 10.0.0.3
        number = 0
        name = cl01
        cluster = ocfs2
node:
        ip_port = 7777
        ip_address = 10.0.0.4
        number = 1
        name = cl02
        cluster = ocfs2
cluster:
        node_count = 2
        name = ocfs2
```

Abbildung 18: OCFS2 Konfigurations-Datei

Nach dem Laden der OCFS2-Module und starten der OCFS2-Cluster-Software erzeugt das Kommando

```
cladmin@cl01~$ sudo mkfs.ocfs2 /dev/drbd0
```

ein Dateisystem auf dem DRBD-Gerät.

Der Einhängepunkt für das Dateisystem wird mit dem Kommando

```
cladmin@cl01~$ sudo mkdir /mnt/ocfs2
```

erzeugt.

Dieses Gerät lässt sich nun mit dem Kommando

```
cladmin@cl01~$ sudo mount -t ocfs2 /dev/drbd0 /mnt/ocfs2
```

in das System einhängen.

Allerdings schlugen Versuche das Dateisystem auch auf dem zweiten Knoten einzuhängen fehl und führten zu Systemabstürzen.

5.4.2.2 Implementierungsversuch von OCFS2 über ein iSCSI-Gerät

Bei der Recherche nach möglichen Lösungen für die oben dargestellte Problematik warf [FAQ-2007] einen alternativen Ansatz zur Realisierung auf.
Da DRBD zurzeit die SCSI-Semantiken zum Betrieb zweier primärer Knoten scheinbar noch nicht in der nötigen Qualität unterstützt, wird dort eine Lösung auf Basis von iSCSI empfohlen. Bei iSCSI handelt es sich um Protokolle, mit denen SCSI-Kommandos über IP-Netze übertragen werden können [SAT-2004].
Der hier gezeigte Ansatz verfolgt die Idee, ein DRBD-Gerät mit nur einem primären Knoten zu verwenden. Auf dieses DRBD-Gerät setzt ein iSCSI-Target auf. Dieses kann von mehreren iSCSI-Initiatoren, in diesem Fall beide Cluster-Knoten, angesprochen werden und erscheint auf der Seite des Initiators als lokale Festplatte. Auf dieser wird nun ein OCFS2-Dateisystem angelegt und auf beiden Knoten eingehängt. Das DRBD-Gerät und das iSCSI-Target bilden Heartbeat-Ressourcen.
Beide iSCSI-Endpunkte können mit Open-Source-Software realisiert werden. Als Initiator steht open-iscsi als fertiges Paket zur Verfügung. Ein iSCSI-Target lässt sich mit der Software des Projekts *iSCSI Enterprise Target* (http://iscsitarget.sourceforge.net/) realisieren. Diese liegt nur im Quellcode vor und muss auf dem Zielsystem übersetzt werden.
Damit ließ sich ein iSCSI-Target auf das DRBD-Gerät abbilden. Auch die Verbindung der iSCSI-Initiatoren auf das Target war realisierbar, genau wie das Anlegen eines OCFS2-Dateisystems von einem Knoten aus.
Allerdings kam es auch hier wieder zu Systemabstürzen des zweiten Knotens beim Zugriffsversuch auf das OCFS2-Dateisystem.

5.5 Konflikte mit dem Shared-Nothing-Konzept von MySQL

Im Gegensatz zum bisher verfolgten Ansatz des Shared Storage verfolgt MySQL das Konzept des Shared Nothing zur Realisierung von Clustern. In einem Shared Nothing

Cluster existiert kein gemeinsamer Speicher, was die Notwendigkeit aufwendigen Lock-Managements beseitigt. Stattdessen verfügt jeder Knoten über seinen eigenen Speicher und die Software selbst muss über Mechanismen zur Synchronisation und Sicherung der Datenkonsistenz verfügen.

Das Shared Nothing Konzept als solches stellt kein technisches Hindernis für den Einsatz in der hier dargestellten Umgebung dar. Das Hindernis besteht darin, dass laut [DAV-2006] für den Einsatz eines MySQL SharedNothing Clusters mindestens drei Knoten benötigt werden. Dies ist mit einem Zwei-Knoten-Cluster nicht realisierbar.

6 Zusammenfassung

Mit den Ergebnissen, die in dieser Arbeit erzielt wurden, ist es möglich redundante VoIP-PBX-Lösungen aus Basis von Open-Source-Software mit Standard-Hardware zu realisieren. Dazu wird ein HA-Cluster bestehend aus zwei Knoten aufgebaut, das nach außen als ein einzelnes System erscheint. Intern verfügt es mit Heartbeat und DRBD über Technologien, durch die selbst ein Totalausfall eines Knoten nicht nach außen sichtbar ist.

Ebenso wurden für alle geforderten Arten von PSTN-Anbindungen Redundanzlösungen skizziert, die teilweise spezielle Hardware-Komponenten voraussetzen.

Basierend auf diesen Erkenntnissen wurde das System um einen Netzwerk-Load-Balancer in einer streamline-Konfiguration erweitert. Damit ist es nun möglich, die Client-Anfragen auf beide Knoten zu verteilen und dort abzuarbeiten, ohne Einbußen bei der Ausfallsicherheit des Clusters hinnehmen zu müssen.

Die konsequente Weiterführung dieses Konzepts machte den Einsatz eines Cluster-Dateisystems auf dem Shared Storage notwendig, um beiden Knoten den simultanen Zugriff auf relevante Daten zu ermöglichen. Dies scheiterte in der praktischen Umsetzung zu diesem Zeitpunkt, da sich die Arbeit hier auf Höhe der aktuellen Entwicklung im Open-Source Bereich befindet. Dadurch lagen nicht alle benötigten Software-Komponenten in ausreichend stabilen Versionen vor.

7 Ausblick und Weiterentwicklung

Ausgehend von den in Kapitel 5.4.2 gewonnen Erkenntnissen lassen sich einige Ansatzpunkte für die Weiterentwicklung der in dieser Arbeit behandelten Thematiken formulieren.

Die in dieser Arbeit vorgestellten Technologien und Open-Source-Prjekte wie Heartbeat, DRBD, ipvs, den Cluster-Dateisystemen und iSCSI eignen sich nicht nur für den Aufbau von VoIP-HA-Clustern. Eine Vielzahl von Server-Diensten lässt sich damit redundant auslegen und somit in der Qualität verbessern. Dies ist auch mit kostengünstiger Hardware und ohne zusätzliche Kosten für Software realisierbar. HA- und Load-Balancing-Cluster dieser Art werden schon vielfach eingesetzt und ihre Verbreitung wird vermutlich noch zunehmen.

Besonderes Augenmerk sollte auf die weitere Entwicklung von DRBD und der Cluster-Dateisysteme gelegt werden. Diese weitere Entwicklung ist besonders für Anwendungen mit großen Datenmengen wie beispielsweise große Datei-Server interessant. Die Datenbank-Problematik stellt einen weiteren Punkt zur zukünftigen Bearbeitung dar. Vielleicht lässt sich auch der Ansatz des Sharad-Nothing-Clusters weiter verfolgen.

Ein weitere Ansatzpunkt zur Weiterentwicklung ergibt sich bei dem Produkt independent. Ausgehend von den Ergebnissen dieser Arbeit kann nun die Integration einer HA-Cluster-Komponente in das Produkt geplant werden. Neben der Integration der entsprechenden Pakete in das Produkt muss auch eine benutzerfreundliche Administrationsmöglichkeit geschaffen werden. Hier bietet sich eine Erweiterung der bestehende webbasierten Administrations-Oberfläche um entsprechende Module zur Konfiguration der Cluster-Knoten an.

Um die Leistungssteigerung durch das Load-Balancing nachweisen zu können, wäre der Aufbau einer entsprechenden Testumgebung und die Erarbeitung einerpassenden Messmethode notwendig.

Stichwortverzeichnis

Literaturverzeichnis

ACK-2007: Ackermann, Ralf; Dittler, Hans Peter: *IP-Telefonie mit Asterisk*. dpunkt.verlag, Heidelberg, 2007.

USG-2007: Ubuntu Documentation Project: *Ubuntu Server Guide*. https://help.ubuntu.com/6.06/pdf/ubuntu/C/serverguide.pdf, zuletzt besucht am 10.10.2007.

CLU-2007: Wikipedia: *Computer cluster*. http://en.wikipedia.org/wiki/Computer_cluster, zuletzt besucht am 07.10.2007.

ROB-2000: Robertson, A. L.: *Linux-HA Heartbeat Design*. http://www.linuxshowcase.org/2000/2000papers/papers/robertson/robertson.pdf, zuletzt besucht am 07.10.2007.

ROB-2007: Robertson, A. L.: *The High Availability Linux Project*. http://www.linux-ha.org/, zuletzt besucht am 07.10.2007.

REI-2000: Reisner, Philipp: *DRBD, Festplattenspiegelung übers Netzwerk für die Realisierung hochverfügbarer Server unter Linux*. Technische Universität Wien, Wien, 2000

ZHA-1999: Wensong Zhang, Shiyao Jin, Quanyuan Wu: *Creating Linux Virtual Servers*. http://www.linuxvirtualserver.org/clvs.ps.gz, zuletzt besucht am 07.10.2007.

BAD-2001: Bader, David; Pennington Robert: *Cluster Computing: Applications*, The International Journal of High Performance Computing, May 2001.

LIN-2005: Lindheim, Jan: *Building a Beowulf System*. http://www.cacr.caltech.edu/beowulf/tutorial/building.html, zuletzt besucht am 07.10.2007.

SUN-1990: Sunderam, V. S.: *PVM: A Framework for Parallel Distributed Computing*. http://portal.acm.org/citation.cfm?id=95327&dl=ACM&coll=GUIDE, zuletzt besucht am 08.10.2007.

GAB-2004: Edgar Gabriel; Graham E. Fagg; George Bosilca; Thara Angskun; Jack J. Dongarra; Jeffrey, M. Squyres; Vishal Sahay; Prabhanjan Kambadur; Brian Barrett; Andrew Lumsdaine; Ralph H. Castain; David J. Daniel; Richard L. Graham; Timothy S. Woodall: *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation*. http://www.open-mpi.de/papers/euro-pvmmpi-2004-overview/euro-pvmmpi-2004-overview.pdf, zuletzt besucht am 08.10.2007.

AMA-2007: Aman – Kraehenbuehl, Dr. Daniel Ali: *Asterisk-Cluster based onTrixbox 1.2.3*. http://www.danielaliaman.com/blog/files/phonecube/cluster/AsteriskCluster.pdf, zuletzt besucht am 09.10.2007.

ZAB-2004: Zabawskyj et. al.: *Method and system allowing for one mobile phone number (MSISDN) to be associated with a plurality of wireless devices ('Multi-SIM')*. http://www.freepatentsonline.com/20040229601.pdf, zuletzt besucht am 09.10.2007.

MIL-1997: Milz, Harald: *Linux High Availability HOWTO*. http://www.vergenet.net/linux/redundant_content/related/linux-ha/High-Availability-HOWTO-8.html#ss8.6, zuletzt besucht am 10.10.2007.

TAN-1995: Tanenbaum, Andrew S.: *Moderne Betriebssysteme*. Hanser, München, 1995.

SPE-2004: Spencer, Marc: *Distributed Universal Number Discovery(DUNDiTM) and the General Peering Agreement (GPATM)*. http://www.dundi.info/dundi.pdf, zuletzt besucht am 14.10.2007.

RIC-2006: Richardson, J. R.: *Using DUNDi with a Cluster of Asterisk Servers*. http://www.voip-magazine.com/content/view/3644/, zuletzt besucht am 01.07.2007.

ELL-2007: Ellenberg, Lars: *DRBD 8.0.x and beyondShared-Disk semantics on a Shared-Nothing Cluster*. http://www.drbd.org/fileadmin/drbd/publications/drbd8.linux-conf.eu.2007.pdf, zuletzt besucht am 14.10.2007.

SEI-2007: Seidel, Udo: *Server-Traube, GFS2 und OCFS2, zwei Cluster-Dateisysteme im Linux-Kernel*. http://www.linux-magazin.de/online_artikel/gfs2_und_ocfs2_zwei_cluster_dateisysteme_im_linux_kernel?category=349, zuletzt besucht am 14.10.2007.

FAQ-2007: Linux-HA project: *Frequently Asked Questions about DRBD*. http://linux-ha.org/print.php/DRBD/FAQ, zuletzt besucht am 14.10.2007.

SAT-2004: Satran J.; Sapuntzakis C.; Chadalapaka M.; Zeidner E.: *Internet Small Computer Systems Interface (iSCSI)*. http://www.ietf.org/rfc/rfc3720.txt, zuletzt besucht am 14.10.2007.

DAV-2006: Davies, Alex; Fisk Harrison: *MySQL® Clustering*. http://safari.oreilly.com/0672328550/ch03#X2ludGVybmFsX1RvYz94bWxpZD0wNjcyMzI4NTUwL3ByZWYwNA==, zuletzt besucht am 14.10.2007.