Fakultät Informatik, Mathematik und Naturwissenschaften

Studiengang

Informatik, Master

wissenschaftliche Hausarbeit

im Fach

Hochgeschwindigkeitsnetzwerktechnologien

zum Thema

Peer-to-Peer-Topologien und

Spezifikation einer Peer-to-Peer-Daten-Cloud

Vorgelegt von Alexander Kern

Abgabe am 15.08.2013

Inhaltsverzeichnis

Verschwiegenheitsvereinbarung

Diese wissenschaftliche Hausarbeit ist nur bestimmt für Professor Dr. Hänßgen und Alexander Kern. Eine weitere Verbreitung ist untersagt!

1. Anfang

1.1. Einleitung

Peer to Peer Systeme gibt es bereits eine längere Zeit. In den 1970er Jahren hatte IBM eine logische Einheit von 6 Systemen als Netzwerkarchitektur, die eine Peer-to-Peer-Anwendungskommunikation unterstützt.[Dmitry Korzun(2013)] Die aufkommende Filesharing-Welt kurz vor der Jahrtausendwende entwickelte unstrukturierte P2P-Systeme (P2P=Peer to Peer), die es erlaubten dezentral symmetrische Kommunikation zwischen Uploadern und Downloadern zu ermöglichen.

Napster (1999) gilt in einiger Literatur als das erste P2P-Netzwerk.[Dmitry Korzun(2013)] Jedoch verwendet Napster auch das Server-Client-Paradigma. Die Downloads und Uploads finden zwischen den Peers bzw. Clients statt, aber anmelden muss man sich an einen zentralen Server. Darauf folgte Gnutella (2000), das das P2P-Paradigma erstmals vollständig umgesetzt hat. Man braucht jedoch zunächst eine Liste von Peers, die auf einer extra Internetseite aufzufinden war. Dieser Vorgang nennt sich Bootstrapping. In der P2P-Cloud, um die es in der Arbeit geht, muss dieser Vorgang automatisch erfolgen, was rechtlich, im Gegensatz zu Gnutella, kein Problem darstellt. Aus den Peers und dessen Verbindungen ergibt sich unter Gnutella eine ungeordnete Verbindungs-Struktur des Overlay-Netzwerkes. Das ist insofern relevant, weil so P2P-Netze grundsetzlich unterteilt werden, in geordnete und ungeordnete Netzwerkstrukturen. Overlaynetzwerke sind Netzwerke innerhalb anderer Netzwerke, den Underlay-Netzwerken. Neuere P2P-Netze als Gnutella haben als Overlaynetzwerkstruktur eine Datenstruktur, wie z.B. verteilte Hashtabellen (distributed Hashtabels - DHT's). Dazu müssen alle Peers zusammen diese Struktur aufbauen.

Wenn Netzwerkspezifikationen erstellt werden, müssen alle denkbaren Fragen zur Sicherheit einbezogen worden sein. Eine Hürde dabei ist alle zu entdecken. Wenn ein sicherheitsrelevantes konzeptionelles Problem vorliegt, kann die Spezifikation zum Scheitern verurteilt sein oder das Gesamtkonzept muss von vorn neu konstruiert werden. Dabei spielen nicht nur Gefahren durch Malware (= schädliche Software), Hacker und Cracker, sondern auch Verschulden oder Versagen auf Seiten der Nutzer und Entwickler eine Rolle.
Peers dürfen nicht aus der gemeinsamen Arbeitsweise ausbrechen, was ein Sicherheitsproblem darstellt (gecrackte Peers). Zur einzelnen Bevorteilung, bezogen auf Ressourcen und Daten, kann es lohnenswert sein, einen gecrackten Peer zu verwenden, es sei denn Peers sind vornherein so konstruiert maximal quasi egoistisch zu sein. Da Verbindungen in Gnutella beliebig also ohne System in die Netzwerkstruktur angelegt werden, ist für einen Hacker ein lohnenswerter Angriffspunkt in der Netzwerkstruktur kaum erkennbar. Deshalb ist das Netzwerk robust gegenüber z.B. Überlastungsangriffen. In geordneten Netzwerken lassen sich Knoten finden, an denen ein Überstlastungs-Angriff (z.B. DDOS) erfolgreicher stattfinden kann als an anderen Knoten. In einem chaotischen Netzwerk sind solche Knoten schwieriger aufzufinden. Nachteilig ist, dass nicht alle Dateien im Netzwerk gefunden werden können, da es eine Beschränkung der Suchtiefe in der Tiefensuche im Netzwerk wegen Entlastung gibt. Ein geordnetes Netzwerk lässt sich in der Regel leichter als ein chaotisches Netzwerk einteilen in sensible und robuste Knotenbereiche, bezogen auf Angriffe jeglicher Art.

1.2. Motivation der Konzeption einer P2P-Cloud

Es gibt Dropbox als verbreitete Cloudlösung, Owncloud als Open-Source-Cloud zur Installation auf Server und dutzende weitere vergleichbare Cloud-Lösungen, abgesehen von den altbekannten E-Mail-Diensten. Gemeint sind Clouds, zum Lagern von Daten / vollautomatischen synchronisieren / Datenaustausch mit anderen Accounts. Dies geschieht mit Mechanismen, die eine höhere Bequemlichkeit ermöglichen als sich darauf zu beschränken einen Dateimanger zu verwenden, z.B. der Verwendung von Kernel-

File-Hooks, Benachrichtigungen vom Kernel, dass soeben eine spezielle Änderung im Ordner stattfindet. Da das Vermieten von entfernten Speicherplatz ein lohnenswertes Geschäftsmodell ist, basieren duzende Clouds auf dem Server-Client-Prinzip. Ein verbreitetes Geschäftsmodell ist es beschränkten Speicherplatz kostenlos anzubieten, um Kunden anzulocken und mehr Speicherplatz für Geld bereitzustellen. Die zahlungsbereiten Kunden subventionieren somit die Nutzer der kostenlosen beschränkten Variante.

Es steht dem Kunden frei, seine Daten mit Lösungen von Drittanbietern zu verschlüsseln. Seit 2013 ist bekannt, was lange vermutet wurde, dass Geheimdienste präventiv auf private Daten zugreifen.

Jeder, der eine Daten-Cloud nutzen möchte, aber selbst freien Speicherplatz zur Verfügung hat, kann doch als Cloud den freien Speicherplatz eines anderen nutzen, der widerum den freien Speicherplatz nutzen kann, den man selbst nicht braucht. Das ist die Grundidee der Lösung, um die es in dieser Arbeit geht.
Eine Peer-to-Peer-Cloud, die Redundanzkonzepte des RAID und Kryptographie nutzt, kann dafür eingesetzt werden.

Beispielsweise kann jeder Peer 10 GB Clouddatenspeicher nutzen, wenn er 100 GB für beliebige andere Nutzer frei zur Verfügung stellt. Wenn jeder Peer so arbeitet, ist 10-fache Redundanz pro der 10 GB Clouddatenspeicher für jeden Peer verfügbar. Angenommen es gibt 10 solche Peers, dann sind 10*10GB 10 Fach redundant auf 10*100GB speichern (10*10GB*10 = 10*100GB). Weil Peers offline gehen können und wegen anderen Risiken ist Redundanz in P2P-Netzen wünschenswert und in einem Teil der P2P-Lösungen umgesetzt.

Jeder Peer kann auf die Daten anderer Peers auf seinem Rechner zugreifen. Jemand der einen Peer betreibt, sollte daher seine Daten in der Cloud verschlüsseln. Deshalb macht es Sinn Verschlüsselung in der P2P-Cloudlösung zu integrieren, um die es hier geht.

Diese Arbeit beschränkt sich auf die Topologiewahl und ersten konzeptionellen Spezifikationen, die nicht weiter vertieft werden, weil das den Rahmen sprengen würde.

Verschlüsselung wird also in die Spezifikation aufgenommen, jedoch nicht weiter vertieft.

1.3. Problemstellung der Arbeit

Diese Arbeit untersucht in erster Linie zunächst vorhandene P2P-Topologien, um darauf aufbauend eine dieser für die zu spezifizierende P2P-Cloud zu wählen. Es wird ein Einblick in eine überschaubare Topologie gegeben, damit ein Verständnis in die Problematik möglich ist. Die ausgewählte Topologie muss beschrieben werden und es muss begründet werden, warum sich dafür entschieden wurde. Der Weg zur Spezifikation muss mit Vorüberlegungen geebnet werden, worauf eine Spezifikation folgen muss, die sich auf Kernbereiche konzentriert und außerdem nicht den Anspruch auf Vollständigkeit haben muss, da im Verlauf durch die Implementation und die vergangene Zeit Änderungen an der Spezifikation durchgenommen werden können. Die Kernbereiche betreffen grundlegende Fragen zur Realisierbarkeit, zur Definition der Grundfunktionalität und des Aufbaus des Codes. Vor- und Nachteile im Vergleich zu Alternativen sind zu benennen.

1.4. Aufbau der Arbeit

Der Mittelteil der Arbeit besteht aus 2 Bereichen, 1. der Auswahl der Topologie (2. Peer-to-Peer-Topologien mit Fakten) und 2. der ersten Spezifikation der P2P-Cloud (3. Die P2P-Cloud). Zunächst wird auf die Topologie des P2P-Netzes CAN näher eingegangen (2.1. Verteilte Hashtabellen bei CAN) , um zunächst einen anschaulichen Einblick in die Problematik der Topologien von P2P-Netzen zu gewinnen. Darauf folgt ein Abschnitt der sich mit strukturellen Angelegenheiten bezogen auf die Geschwindigkeit und Effizienz befasst (2.2. Ausgesuchte Topologie-Infos zu vorhandenen Peer-to-Peer-Systemen). Diese werden betrachtet, weil eine Topologie ausgewählt wird, die die Anforderungen am Besten erfüllt.

Im dritten Kapitel “Die P2P-Cloud” wird nun zunächst die ausgewählte Topologie Skipnetz betrachtet (3.1. Skipnetze) und begründet warum sich dafür entschieden wurde. Darauf folgen als nächster Schritt Überlegungen (3.2. Vorüberlegungen zur Umsetzung einer P2P-Cloud) , die in die Spezifikation münden werden, anstelle sofort mit der Spezifikation zu beginnen. Als erste Spezifikation folgt darauf ein UML-Anwendungsfalldiagramm (3.3. UML-Anwendungsfalldiagramm), das zeigt was ein Peer für Handlungsspielräume haben kann. Da das Backend aufwändiger zu programmieren ist und die hauptsächliche Komplexität aufweist, als das Frontend, wird als nächstes spezifiziert (3.4. Hintergrundarbeiten eines Peers). Wie das Frontend gestaltet ist, hat zunächst niedrigere Priorität. Darauf folgen Entscheidungen zur softwaretechnischen Architektur (3.5. Architektur). Da die Daten eines Peers in der Cloud der anderen Peers zeitnah gefunden werden müssen, beschäftigt sich der nächste Abschnitt mit fragen zur Suche im P2P-Netz (3.6. Suchstrategie). In den 3.7. “Gestaltungsrichtlinien der P2P-Cloud” wird beschrieben, welche Ideale die Umsetzung des Produktes charakterisieren muss.

Am Ende 4. “Schluss” folgt die 4.1. “Ergebnisse und Untersuchungen”, darauf 4.2. “wichtige Ergebnisse und Aussagen” und als letztes 4.3. “Ausblick und Alternativen”.

1.5. wichtige Ergebnisse und Aussagen dieser Arbeit

Die gewählte P2P-Netz-Topologie bzw. Datenstruktur ist der Skipgraph, dh. das P2P-Netz ist ein Skipnet Das Suchen nach einem Peer darin benötigt logarithmische Laufzeit. Skipnetzverbindungsstrukturen lassen sich bei Ausfall von Peers besser als P2P-Netze reparieren, die auf bekannten Hash-Verfahren beruhen. Dem Risiko des Datenverlustes und dem Problem der Verfügbarkeit wird entgegnet mit 10-facher oder individueller Redundanzverteilung der Daten eines Peers auf (10) fremden Peers, P2P-Netz-Monitoring, sozialen Bindungen mittels Chat, automatischen, manuellen Bewertungen und Hierachiebenen von Peers. Automatische Verschlüsselung der Daten auf fremden Peers wird notwendig, da diesen in der Regel weniger vertraut wird als Servern von Unternehmen. Eine Hierarchieordnung von Peers kann sich realisieren lassen nach den Ebenen des Skipnetzes. Dutzende (P2P-)Netzwerksimulatoren existieren verschiedener Art (auch bezüglich

Netzwerklayerebene), die im Entwicklungsprozess nützlich sind. Fragmentierung wie bei RAID 0 eignet sich zur Beschleunigung des Datendurchsatzes insbesondere beim Einsatz von asymmetrischem DSL (niedrigere Upload- als Download-Rate).

2. Peer-to-Peer-Topologien mit Fakten

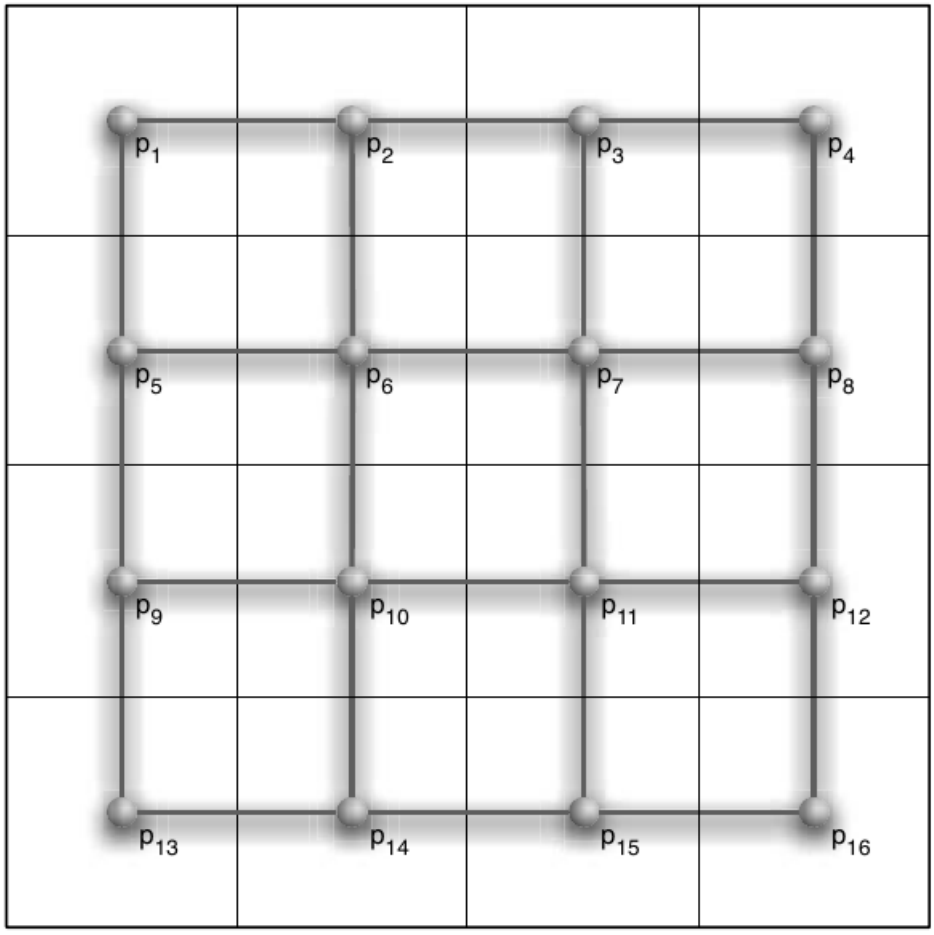
2.1. Verteilte Hashtabellen bei CAN

Im Jahr 2000 erschien CAN (Scalable Content Addressable Network) als skalierendes Netzwerk mit adressierbaren Inhalten. In der Datenstruktur von CAN, aus der dessen Netzwerk besteht, sind die Peers in einem Gitter verbunden. CAN verwendet verteilte Hashtabellen. Diese können in unterschiedlichen Topologien vorliegen. Auch wenn die Hashtabellen in Form einer Gitterstruktur in CAN auftreten, so ließen sich Hashtabellen auch in anderen Formen realisieren, z.B. als Ring mit zusätzlichen Verzweigungen, die einer Ordnung unterliegen, oder als Gitter mit einer Dimension höheren Grades als 2.

In Hashtabellen berechnet eine Funktion zu einem Schlüssel den passenden Index in der Hashfunktion. Im Idealfall lassen sich so in der Laufzeit O(1) Werte finden. Da diese Hashfunktion bei unterschiedlichem Schlüssel den gleichen Index berechnen kann (Kollision) verkompliziert sich die Funktionsweise von Hashtabellen. In dem Fall müssen Verweise in der Tabelle auf andere freie Elemente erstellt werden. Die Tabelle ist dabei immer größer als der Datenbestand, um Kollisionen vorzubeugen und Laufzeiten von O(1) zu gewährleisten.

Die Netzwerkstruktur von CAN ist schematisch ein Quadrat, das x- und y-Koordinaten aufweist siehe Abbildung 2.1. Rechtecke in diesem Quadrat entsprechen Peers. Als ideale Struktur wird das Netzwerk angesehen, wenn alle Rechtecke Quadrate sind, so dass das ganze Netzwerk quasi wie ein Schachbrett aussieht wie auch in Abbildung 1.1. Mittels 2 verschiedener Hashfunktionen wird eine x- und y-Koordinate zugewiesen zu einem Peer. Die Peers sind bezüglich deren Rechteck und zugehörigen Daten autark. So ist bekannt, welcher Peer welche Daten speichert. Wenn sich nun ein neuer Peer in die

Abbildung 2.1.: Die Netzwerkstruktur von CAN in idealer Anordnung der Peers[Mahlmann(2007)]



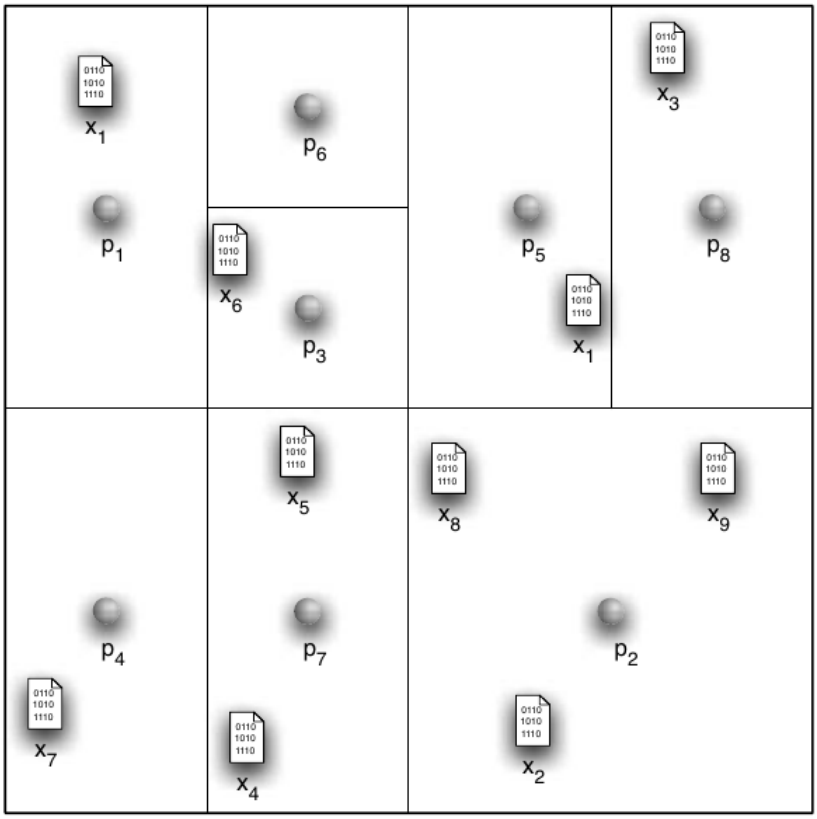
Netzwerkstruktur einbindet, wird ein Rechteck in der Mitte halbiert, so dass der alte und neue Peer innerhalb des alten Rechteckes vorliegen. Die Eigenschaft, dass andere Peers davon unbeeinflusst sind nennt sich Konsistenz. Im Idealfall haben alle Peers gleich viel Platz in dem Quadrat. Der Peer mit dem zugehörigem Rechteck, siehe Abbildung 1.2, kommuniziert mit den Peers in den zugehörigen Rechtecken links, rechts, oben, unten vom eigenen Rechteck, aber nicht quer.[Mahlmann(2007)]

Durch das andauernde Löschen und Hinzufügen von Peers entfernt sich das Quadrat der Rechtecke, also dieser Topologie, zunehmend von der idealen Struktur eines Quadrates aus ausschließlich Quadraten (ideal: Abbildung 2.1 , nicht ideal Abbildung 2.2). Die Ordnung der Netzwerkstruktur fragmentiert also. Damit sich dies nicht nachteilig auf die Leistungsfähgikeit auswirkt gibt es in regelmäßigen zeitlichen Abständen eine Defragmentierung der Struktur des Overlaynetzwerkes. Das ist eine Umstrukturrierung von einem nicht idealem Zustand in Richtung idealem Zustand. In der Literatur ist dazu kein Algorithmus angegeben. Es wird bei diesem Vorgang angestrebt, dass weniger längliche Rechtecke und mehr Quadrate vorliegen und die Größe aller Rechtecke möglichst gleich ist. Ein Peer kann daher auch für mehrere Rechtecke verantwortlich sein. In der Defragmentierung wird CAN als Binärbaum betrachtet. [Mahlmann(2007)]

Es gibt Möglichkeiten CAN effizienter zu gestalten. Es ist beispielsweise möglich anstelle von einem Quadrat als Netzwerk von einem mehrdimensionalen Würfel auszugehen. Es erhöht sich so der Durchmesser auf $O(n^{1/d})$ aus Graphentheoretischer Sicht (d ist der Grad der Dimension des Quadrates / Würfels / .... und n dessen Ausdehnung), und einem Grad von O(d) bei gleichgroßen Hyper-Rechtecken.[Mahlmann(2007)]

Eine weitere Möglichkeit ist es, für alle Netzwerkteilnehmer zusätzliche Overlay-Netzwerke aufzuspannen. Diese werden dann als Realitäten bezeichnet. Dabei ändert sich der Durchmesser eines CAN. Eine weitere Möglichkeit ist es, einem Rechteck mehr als einen Peer zuzuweisen, wobei in diesem Rechteck alle Kontakte zu den rechten, linken, oberen und unteren Peers für alle Maschinen des zugehörigen Rechteckes die gleichen Peers sind. Ein Vorteil dabei besteht, dass man im Rechteck einen Peer für die Kommunikation mit den Nachbarpeers wählt, der geringe Latenzen besitzt. Dies beschleunigt das

Abbildung 2.2.: Die Netzwerkstruktur von CAN in realer und nicht idealer Anordnung der Peers[Mahlmann(2007)]



Routing.[Mahlmann(2007)]

Das Overlaynetzwerk eines P2P-Systems ist gewöhnlich unabhängig von den Strukturen des Internets. So kann ein Peer, der direkt mit einem anderen Peer verbunden ist, auf einem anderen Kontinent sitzen. Von daher gibt es die Option, das Overlaynetzwerk des P2P-Systems durch Defragmentier- und Fragmentier-funktionen dem Internet anzupassen, um eine Optimierung der internen Kommunikation zu erreichen. Ein Ziel dabei kann es sein, dass Peers, die im Internet verhältnismäßig nah verbunden sind im Overlaynetzwerk Nachbarn werden. Dies kann sowohl in regelmäßigen zeitlichen Abständen umgesetzt werden als auch bei Verbindungsbeginn.[Mahlmann(2007)]

Eine Möglichkeit, die Robustheit zu erhöhen ist es, Daten mehrfach zu platzieren. Dies reduziert die Anzahl der Zwischenstationen für eine Suche und die Latenzzeiten.[Mahlmann(2007)]

2.2. Ausgesuchte Topologie-Infos zu vorhandenen Peer-to-Peer-Systemen

In diesem Abschnitt werden einige Informationen zu verbreiteten P2P-Netzen genannt. Dazu gehören Laufzeiten von Operationen auf diesen Netzen, die zugrundeliegende Datenstruktur des Overlaynetzwerks, Fakten zur Effizienz von Netzwerkoperationen mit Begriffen aus der Graphentheorie, Bilder zu Veranschaulichung der Netzwerkstruktur, Erklärungen zur Funktionsweise der Datenstruktur. Diese Informationen werden zusammengetragen, um einen Überblick zu gewinnen über vorhandene Technologie, um darauf aufbauend Entscheidungen treffen zu können diese auszuwählen oder schöpfend zu rekombinieren, für die Planung der P2P-Cloud. Ein Überblick soll also geschaffen werden. Vertiefungen wären deshalb nicht zielführend.

Das Schema CANs eines Quadrates kann erweitert werden auf einen Kubus oder aus einem Torus, wobei die Peers weiterhin Rechtecken entsprechen. Es handelt sich um kartesische Koordinaten. Die durchschnittliche Pfadlänge beträgt $(\frac{d}{4}(n^{\frac{1}{d}})$ (d = Grad der Dimension ; n = Ausdehnung). Das Hinzufügen von Knoten erhöht die durchschnittliche

Pfadlänge um O(n^(1/d)).[Dmitry Korzun(2013)]

Das P2P-Netz Chord hat eine verteilte Hashtabelle ohne Anfang und Ende, also als Kreis, und mit zusätzliche Verzweigungen innerhalb des Kreises.
Kademlia, ein weiteres P2P-Netz, nutzt auch verteilte Hashtabellen und reduziert die Anzahl der Nachrichten zur Netzwerkverwaltung, die benötigt werden, um die Routingtabelle aufrecht zu erhalten. Knoten lernen über die Peers während der Nachschlage-Operationen. Bittorrent, ein P2P-Protokoll und eine Softwarelösung zum gegenseitigen Austausch von belibigen Daten im Internet (Filesharing), nutzt die Topologie von Kademlia, also s.o. eine Topologie basierend auf verteilten Hashtabellen.[Dmitry Korzun(2013)]

Chord benutzt eine Ringstruktur mit konsistentem Hashing (Eine konsistente Hash-Funktion ist eine Hashfunktion, die die Anzahl der Neuzuordnungen minimiert. Neuzuordnungen erfolgen immer dann, wenn Behälter hinzukommen oder entfernt werden.), eine Successor-List (Successor = Nachfolger in der Liste) und einem Join-Protokoll (Protokoll, um neue Knoten ins Netzwerk hinzuzufügen). Accordion nutzt einen Greedy-Routing-Ansatz. “Greedy” ist in der Algorithmentheorie ein Charakteristikum von Algorithmen. Greedy (gierig) bedeutet, dass der Algorithmus im Ablauf zuerst maximalen Nutzen erreichen will. In manchen Fällen ist das die beste Wahl, in manchen die schlechteste.[Dmitry Korzun(2013)]

O(log N) Knoten müssen kontaktiert werden, um einen Schlüssel in einem Ring von N Knoten zu lokalisieren.[Dmitry Korzun(2013)]

Die P2P-Protokolle und -Netze Pastry und Tapestry verwenden einen Binärbaum als Topologie.[Dmitry Korzun(2013)]

Bamboo ist eine ringbasierte verteilte Hashtabelle (=DHT= distributed Hashtable), mit zusätzlichen Verweisen, die den spezifischen Fokus auf effiziente Operationen in der Anwesenheit von Suchoperationen hat in verschiedenen DHTs und verschiedenen Strategien das Suchen zu beherrschen.[Dmitry Korzun(2013)]

Die Datenstruktur Trie bzw. Präfixbaum hat ihren Name von retrieval (= wieder-auffinden) Abbildung 2.1 : In einem Präfixbaum werden pro Knoten Buchstaben eines Teilwortes gespeichert. Von der Wurzel bis zu einem Blatt kann ein Wort von vorne bis hinten gelesen werden, das sich aus den Wörtern in den Baumelementen zusammensetzt. In der Abbildung 2.3. fängt jedes dieser möglichen Wörter mit “re” an (zweitoberster Knoten), weil es von der Wurzel aus gesehen keine Alternative gibt. Dann kann mit der Nächsten Silbe das Wort “real” / “reb” / “red” / “relie” gebildet werden, wenn als nächstes der Buchstabe “a” / “b” / “d” / “l” folgt, also den ersten Buchstaben der Silbe, die darauf folgt. Das Wort ist erst vollständig, wenn bis zum Blatt weitergearbeitet wurde. Wird z.B. in Abbildung 2.3. immer links entlang gegangen entsteht das Wort “realizes”. In dem Bereich zwischen Wurzel und Blatt kann so entschieden werden, ob das eine oder das andere Wort mit dem gleichen Präfix entsteht, wenn bis zum Blatt gegangen wird.
Ein Präfixbaum definiert die Position in einem Wort innerhalb des Baums. Das Lookup (=Suchanfrage), die Einfügung, die Löschung ist in Teilen der gleiche Quellcode. Das ist ein Argument zur Effizienz gegenüber binären Suchbäumen und Hashtabellen. Das Matching ab dem längsten Präfix oder das Finden eines Knotens mit dem längsten Präfix mittels eines Schlüssels ist effizient mit einem Trie.[Dmitry Korzun(2013)]

Fakten zur Datenstruktur Skipliste:

Die Skipliste findet Anwendung als gewöhnliche Liste. Tatsächlich besteht sie schematisch aus mehreren Schichten von Listen, worauf in Abschnitt 3.1 näher eingegangen wird.

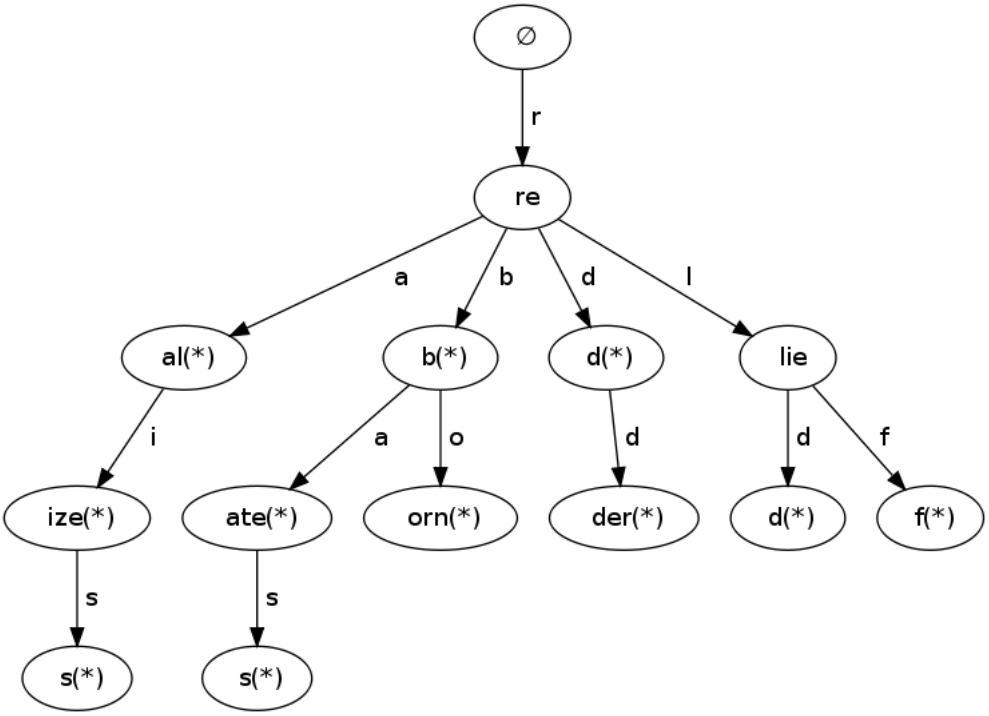
Die Anzahl der Schichten von Dateneinheiten in einer Skipliste beträgt log(1/p).

Die erwartete Suchzeit eines Elementes in dieser Liste beträgt log(1 / p) N) / p.

[Dmitry Korzun(2013)]

Die Topologie vom P2P-Netz Koorde ist eine verteilte Hashtabelle, die insoweit in Chord modifiziert ist, hin zu der Topologie der De-Bruijn-Graphen. Nebenbei erwähnt sei, dass Koorde in der niederländischen Sprache sich wie Chord anhört. Koorde ist fähig, Routing-Anfragen mit sogar 2 Nachbarn pro Knoten in der Laufzeit O(log N) zu bewerkstelligen. Wenn Teilnehmer (Peers) die Anzahl von Nachbarn je eines Peers bzw. Knoten auf die Anzahl log N setzen, sinken die Sprünge einer Anfrage auf O(log N / log log N). Dies erhöht jedoch den Overhead.[Dmitry Korzun(2013)] Weiter vertieft auf den Aufbau und

Abbildung 2.3.: Datenstruktur, der Trie bzw. Präfixbaum, Präfixe führen zum Wort[Dmitry Korzun(2013)]



die Funktionsweise wird u.a. auch hier nicht auf die Struktur von Chord / Koorde und De-Bruijn-Graphen nicht eingegangen, da dies nicht zielführend in der Arbeit ist.

Das P2P-Netz Viceroy verwendet die Butterfly-Topologie mit einer festen Anzahl von Peer-Nachbarn, verwendet z.B. in Multicore-CPUs oder PC-(Overlay)-Netzwerken. Der Knotengrad ist unter O(log N). Der Grad in einem Graphen ist die Anzahl der Kanten, die den Knoten v mit einem anderen Knoten verbinden. Eine Schlinge wird dabei doppelt gezählt. Er gibt die Anzahl der Kanten an, die mit dem Knoten zu anderen Knoten verbunden sind. Die Pfade zeigen, wie die Verbindungen zwischen den Peers vorliegen. Adressiert werden die Knoten mit i und x.[Dmitry Korzun(2013)]

Weitere Punkte sind das Routing und die Aufrechterhaltung der hierarchischen Nachbarn, die zur Vollständikeit hier erwähnt werden, aber nicht zielführender Gegenstand dieser Arbeit sind.[Dmitry Korzun(2013)] Clustering von Peers (Cluster ist hier eine Menge von Peer mit ähnlichen Eigenschaften) kann die Skalierbarkeit vergrößern, hier nur erwähnt für den Fall, dass bei der Konzeption der P2P-Cloud vielleicht darauf zurück

gegriffen wird. Ranking von Peers kann die Kooperation innerhalb eines P2P-Netzes verbessern und wird auch in der Literatur diskutiert. [Dmitry Korzun(2013)]

Es gibt Netzwerksimulationssoftware, z.B.

- OMNeT++ Discrete Event Simulator ist ein diskretes Ereignissimulationstool. Es beinhaltet einen Netzwerkeditor (GNED) zur Erstellung von Netzwerktopologien, eine graphische Simulierungsausführungsumgebung.
- INET Framework for OMNeT++ ist eine Opensource-Zusammenstellung von Modellen, die mit OMNeT++ erstellt wurden.
- OverSim: P2P Overlay Simulation Framework für OMNeT++ , Oversim wurde entwickelt unter dem Rahmen des ScaleNet-Projekts. Es ermöglicht die Simulation von P2P-Overlay-Netzwerken mit OMNeT++ und dem INET-Framework.
- RealPeer
- Es gibt Simulationssoftware auf Sprachenebene. Diese Klasse unterstützt die Implementation des Modells im Modellbildungsprozesses, der Entwicklung eines Simulationsprogrammes.
- Simulationssoftware auf Modellebene: Das sind fertige ausführbare Simulationsprogramme
- Unterstützungssysteme: Dies ist eine integrierte und manchmal interaktive Arbeitsumgebung zur Unterstützung einer Simulationsstudie

[Hildebrandt(2007)]
Diese Lösungen in der Auflistung sind auf Brauchbarkeit zu überprüfen zur Simulation neben der Entwicklung der P2P-Cloud. Dieser Schritt findet jedoch erst statt, wenn 1. anfangen wurde zu entwickeln und 2. Simulationen möglich werden durch Vorhandensein der zunächst halb funktionierenden P2P-Cloud.

Des weiteren gibt es Netzwerk-Simulatoren die physische Netzwerke nachbilden mit jeweiligen Protokollen. Ein P2P-Simulator kann, aber muss nicht, einen Netzwerksimulator

verwenden. Der P2P-Simulator nutzt dabei die ISO OSI Schicht 5-7, wohingegen der Netzwerksimulator die Schichten 1 bis 4 umsetzt.

[Hildebrandt(2007)]

3. Die P2P-Daten-Cloud

3.1. Skipnetze

Skipgraphen bauen auf dem Konzept der Skipliste auf. Skiplisten werden benutzt wie gewöhnliche Listen, haben jedoch einen anderen Aufbau als eine ArrayList oder eine einfach oder einfach / doppelt verketten Liste.

Die untersten Kästchen auf der Abbildung 3.1. entsprechen einer Liste. Diese ist der Träger der nutzbaren Informationen. Der Rest sind strukturriert angeordnete Zeiger. Ein Kästchen, das sich über einem anderen Kästchen befindet ist das gleiche Element wie das darunter, jedoch mit einem weiteren Zeiger. Die Zeiger sind die Pfeile. Wenn eine Suchanfrage durchgeführt wird auf einer Skipliste, wird zunächst das oberste Element auf der Abbildung 3.1. betreten, das vor dem aufzufindenden Element liegt. Eine Ebene darunter wird den dort vorhandenen Zeigern gefolgt bis zum Element, das am dichtesten vor dem gesuchten Element liegt und dann das gleiche gemacht auf der nächst-unteren Ebene, es sei denn das nächste Element rechts ist das gesuchte Element. In dem Fall terminiert der Vorgang erfolgreich. So sind Suchanfragen in logarithmischer Laufzeit (O(log n)) vorliegend.

Die Skipliste ist dabei nicht zu verwechseln mit dem Binärbaum. Die jeweiligen Abstände, ab welcher Größe der Skipliste eine höhere Ebene gebildet wird, oder in welchen Abständen Elemente höherer Ebene vorliegen ist abhängig von der Implementierung. Wenn Elemente in der Liste gelöscht werden oder eingefügt werden, entfernt sich die Skipliste immer weiter von der idealen Struktur, die aus aufeinander aufbauenden Symmetrien besteht.

Die Abbildung 3.2. zeigt einen nicht vollständig idealen Skipgraphen. Der Hauptunter-

Abbildung 3.1.: Die Datenstruktur Skipliste in idealer Form, d.h. mit aufeinander aufbauenden Symmetrien

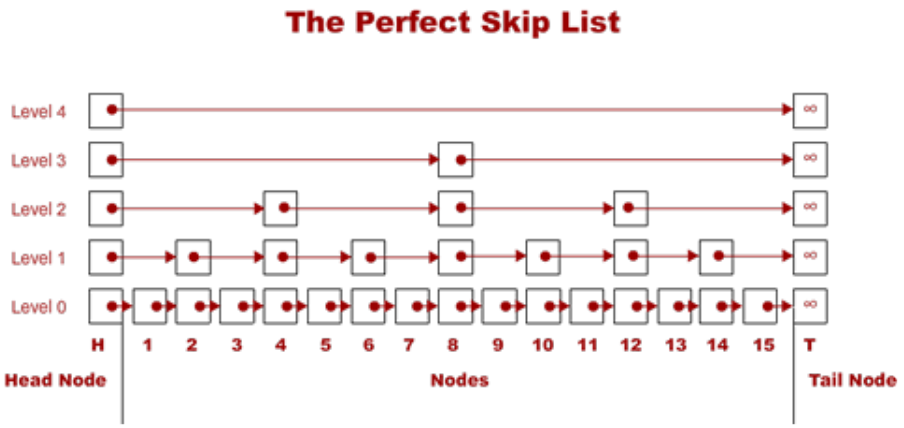
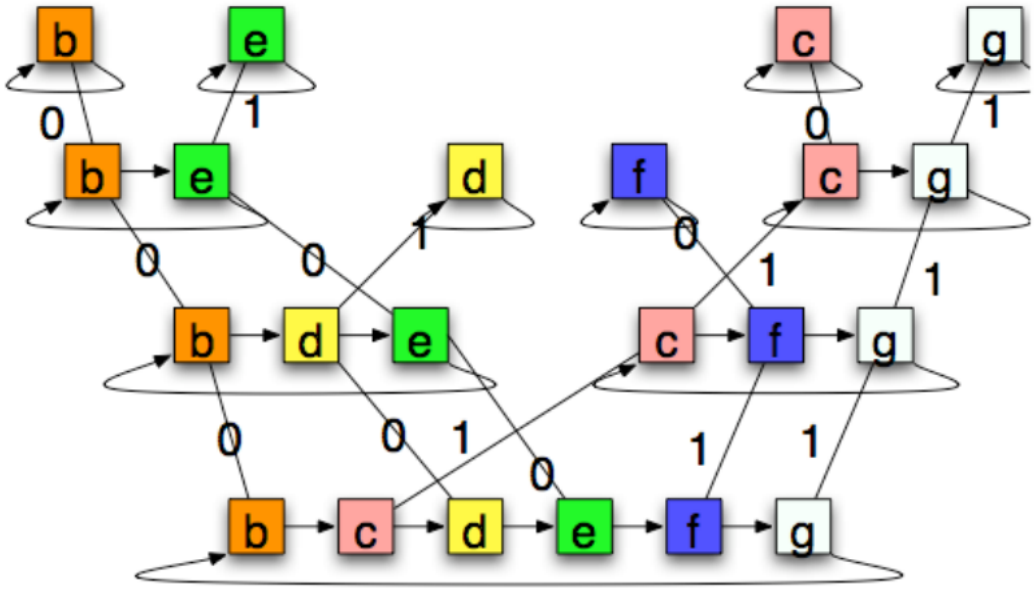


Abbildung 3.2.: Beispiel der Datenstruktur Skipgraph [Dmitry Korzun(2013)]



schied zur Skipliste ist, dass alle Listen und Teillisten nicht mit einem Nullzeiger enden, sondern dass das Ende wieder auf den Anfang zeigt. Sogar Listen, die nur aus einem Element bestehen zeigen auf sich selbst. die oberen Teillisten bestehen nur aus Elementen, die in der untersten Liste vorhanden sind. Suchanfragen erfolgen genauso wie beschrieben in der Skipliste. Alle Listen und Teillisten müssen sortiert vorliegen, z.B. alphanumerisch. Skipnetze sind nun nichts anderes als Skipgraphen als Netzwerkstruktur, wobei die Elemente den Peers entsprechen. Neben einem einfach verketteten Skipnetz ist auch ein doppeltverkettetes denkbar.

Der Vorteil eines Skipgraphen ist dieser,

- dass keine umständliche Balancierung vorgenommen werden muss, wie z.B. bei der Datenstruktur AVL-Baum. Bei der Balancierung werden Teile der Datenstruktur umgeordnet, um die Datenstruktur in einem definierten Zustand zu halten, wenn es zu Einfüge- oder Löschoperationen kommt.
- dass die Daten immer geordnet vorliegen und bleiben,
- dass die Zeit Elemente zu finden O(log n) ist (jedoch haben Hashlisten O(1)),
- die Eigenschaft der weiteren Erreichbarkeit von Knoten im Fall von unkalkulierten Löschungen in der Struktur: Sogar Worst-Case-Fehler können nur einen begrenzten Schaden anrichten.
- Weitere Vorteile sind, dass der Durchmesser durchschnittlich O(log n) beträgt.
- Skip-Graphen sind skalierbar und das Routing ist lokal.
- Ein Annähern an die symmetrisch ideale Struktur und eine Reparatur der Struktur ist möglich. Dazu müssen sich die Peers zueinander abstimmen.
- Die Daten münden nicht in einer Streufunktion, wie beim Einsatz von Hash-Tabellen / DHTs (verteilte Hashtabellen mit jeweiliger Topologie), so dass Bereichsanfragen möglich sind. Das heißt, dass es performant möglich ist auf der Liste, die Elemente in einem Bereich mit einer Suchanfrage zu bestimmen. Wenn Daten redundant gespeichert werden müssen, ist dies daher nützlich diese mit nur einer Suchanfrage

aufzufinden. Ein Multicast ist so zwar nicht umgesetzt, jedoch wäre eine darauf aufbauende Implementierung eines Multicasts performanter als auf einer verketteten Liste.

- Es gibt einen Lastenausgleich. Das heißt die Algorithmen des Skipgraphen sind Parallelisierbar und es können gleichzeitig an verschiedenen Orten im Skipgraphen Veränderungen stattfinden.
- Dezentralisierung wird möglich.

Des Weiteren gibt es deterministische Skip-Netze, die Rotationen einbeziehen. Wenn es eine Abweichung von der idealen symmetrischen Struktur, oder Lücken vorliegen, findet die so genannte Rotation statt. Es findet eine Umordnung der Struktur der Zeiger statt, jedoch nicht der Anordnung der nutzbaren Daten. Der allgemeine Begriff Defragmentierung kommt dafür in Frage. Dadurch kann auf Zustandsänderungen reagiert werden, mit Korrekturen in Richtung idealer Struktur von aufeinander aufbauenden Symmetrien. Diese Rotationen werden in dieser Arbeit nicht algorithmisch ausgeführt.

Ein Nachteil ist, dass das Einfügen von Datenelementen in die Struktur des Skipgraphen $O(log^2(n))$ Zeit benötigt.

3.2. Vorüberlegungen zur Umsetzung einer P2P-Cloud

Die Cloud entspricht hier einer Untermenge fremder Peers. Anforderungen an die Topologie müssen formuliert werden. RAID bietet u.a. redundante Speicherung und / oder schnelleres Lesen und Schreiben. Letzteres wird realisiert dadurch dass hintereinander liegende Daten in kleinste Teile zerlegt werden, die abwechselnd auf mehrere Datenträger verteilt liegen. So wird beim Lesen und Schreiben immer auf alle Datenträger zugegriffen. Im gewöhnlichen Raid werden mehrere Festplatten im Verbund als eine Festplatte verstanden. In der P2P-Cloud, die sich Methoden von Raid bedienen kann, liegen Daten gleichmäßig verteilt auf den Peers, so dass Redundanz so vorliegt, dass das Risiko von Datenverlust maximal verteilt ist.

P2P-Programme, die Dateien zwischen Peers austauschen werden in der Regel für Filesharing verwendet. Das hier zu spezifizierende Programm dient nicht dem Filesharing. Es wird bis einschließlich der ersten Releaseversion unterbunden bzw. nicht spezifiziert, dass Accounts Zugriff haben auf Daten anderer Accounts. Für das Anbieten der Daten eines Accounts ist eine weitere Verschlüsselung notwendig und damit weitere Konzeption notwendig, denn

die Daten des eigenen Peers liegen verschlüsselt auf den entfernten Peers. Dies ist notwendig, da nicht von Gutglauben ausgegangen werden kann, wenn man seine Daten auf fremden Maschinen lagern möchte. Eine weitere Gefahr ist, dass die Besitzer der fremden Computer die Daten löschen. Jedoch existiert dafür Redundanz. Es gibt weitere offene Fragen. Die Frage nach dem Vertrauen zu anderen Peers ließe sich einerseits automatisch lösen, indem festgestellt wird, ob ein Peer einfach Daten gelöscht hat von einem anderen. Dies kann in ein Bewertungssystem einfließen. Andererseits kann eine manuelle Bewertung stattfinden, die mit den anderen Peers abgeglichen wird, da es keinen Server gibt in einem reinen P2P-Netz. Das Funktionieren eines Bewertungssystems innerhalb eines P2P-Netzes ist eine weitere Problematik, für die eine Lösung zu suchen ist. Das ist nicht Gegenstand dieser Arbeit.

Eine entscheidende weitere Frage ist die Wahl der Topologie. Weil die Peers gleichgestellt sind, da es keinen zentralen Server gibt (auch als Meta-Server bekannt, als Verwaltungsinstanz für andere Server und / oder Peers), müssen die Peers selbstständig die Datenstruktur erzeugen. In der vorliegenden Arbeit wurde sich für den Skip-Graph und dementsprechend das Skip-Net entschieden. Wie die Peers zusammen dieses Skipnet aufbauen wird hierbei jedoch nicht beschrieben, da dies den Umfang sprengt.

In einem Skipgraph besteht die Frage, welche Knoten zusätzlich in welcher Schicht des Skipgraphen liegen. Je “höher” der Knoten liegt desto öfter wird dieser von Suchanfragen betroffen sein, als Knoten, die in niedrigeren Ebenen liegen, siehe Abbildung 3.1 und 3.2. Normalerweise werden diese Knoten danach ausgewählt, in welchem Abstand sie im Verhältnis zu den anderen Knoten stehen. Jedoch bietet es sich an ein Bewertungs-

system für solche Knoten zu verwenden. Die Häufigkeit der Onlinezeit, der maximale Datendurchsatz, die Latenz-Zeiten, die tatsächliche räumliche Nähe können in einem Bewertungssystem eine Rolle spielen, das bestimmt wie hoch in den Ebenen des Skipgraphen ein Peer liegen darf. In dem Bewertungssystem müssen nichtlineare Funktionen Einzug halten, denn es muss überlinear (Eine überlineare Funktion ist progressiver im Wachstum, als eine lineare Funktion, z.B. x^2) sichergestellt werden, dass der Aufbau des Skipnetzes gleichmäßig, annähernd symmetrisch ist. Denkbar ist auch ein regelmäßiger Defragmentierungsprozess, um die Skip-Datenstruktur nahe dem idealen Zustand zu halten.

Diese Vorüberlegungen bestehen teils aus Anforderungsanalysen, aber auch darin weiterführenden Designentscheidungen. In der folgenden Spezifikation in der Arbeit wird beschrieben, wie die Designentscheidungen das Backend betreffen. Diese Vorüberlegungen betreffen auch den Handlungshorizont eines Nutzers im Anwendungsfalldiagramm. Es werden in der Benutzerschnittstelle Funktionen nötig sein, die die Nachteile einer P2P-Cloud ausgleichen können. Es ist auch möglich, auf automatische Mechanismen zu vertrauen. Der Handlungsspielraum eines versierten Nutzers, der die Nachteile einer P2P-Cloud kennt wird insbesondere im folgenden Abschnitt 3.3 beschrieben.
Die Vorüberlegungen betreffen auch Architekturentscheidungen, denn File-Hooks lassen sich gut mit dem Beobachter-Entwurfsmuster ansteuern, die in dem Abschnitt 3.5. “Architektur” beschrieben werden.
Die Überlegung zur Defragmentierung in diesem Abschnitt mündet in Lösungsansätzen im Abschnitt der 3.6. “Suchstrategie”

3.3. UML-Anwendungsfalldiagramm

Abbildung 3.3 zeigt ein UML-Diagramm. Von solchen Diagrammen gibt es verschiedene Ausführungen, die namentlich spezifiziert wurden. Anwendungsfalldiagramme bestehen u.a. aus Zielen (was jemand machen will), Akteuren (i.d.R. Programmbenutzer, wer etwas machen will) und Anwendungsfällen einer Software innerhalb von Ellipsen (was gemacht werden kann), So genannte Assoziationen sind die Linien zwischen Akteuren und

Anwendungsfällen. Sie dienen der Darstellung und Planung von Handlungsspielräumen und Aktionen, die Personen in dem zu planenden Softwareprodukt ausführen kann, bzw. können wird. Dies dient der Spezifikation der Anforderungen an das Produkt.

In Abbildung 3.3 ist ein Peer dargestellt mit seinen Handlungsspielräumen. Ansonsten gibt es noch andere Peers, aber für diese gilt das gleiche wie für einen Peer, es sei denn es entstehen Szenarien, in denen es komplexe Situationen mit mehreren Peers gibt. Es ist wünschenswert, wenn diese jedoch automatisch gelöst werden und nicht in Form eines Anwendungsfalles manuell zu erledigen sind.

Abbildung 3.3.: erstes Anwendungsfalldiagramm einer P2P-Cloud aus Sicht eines Peers

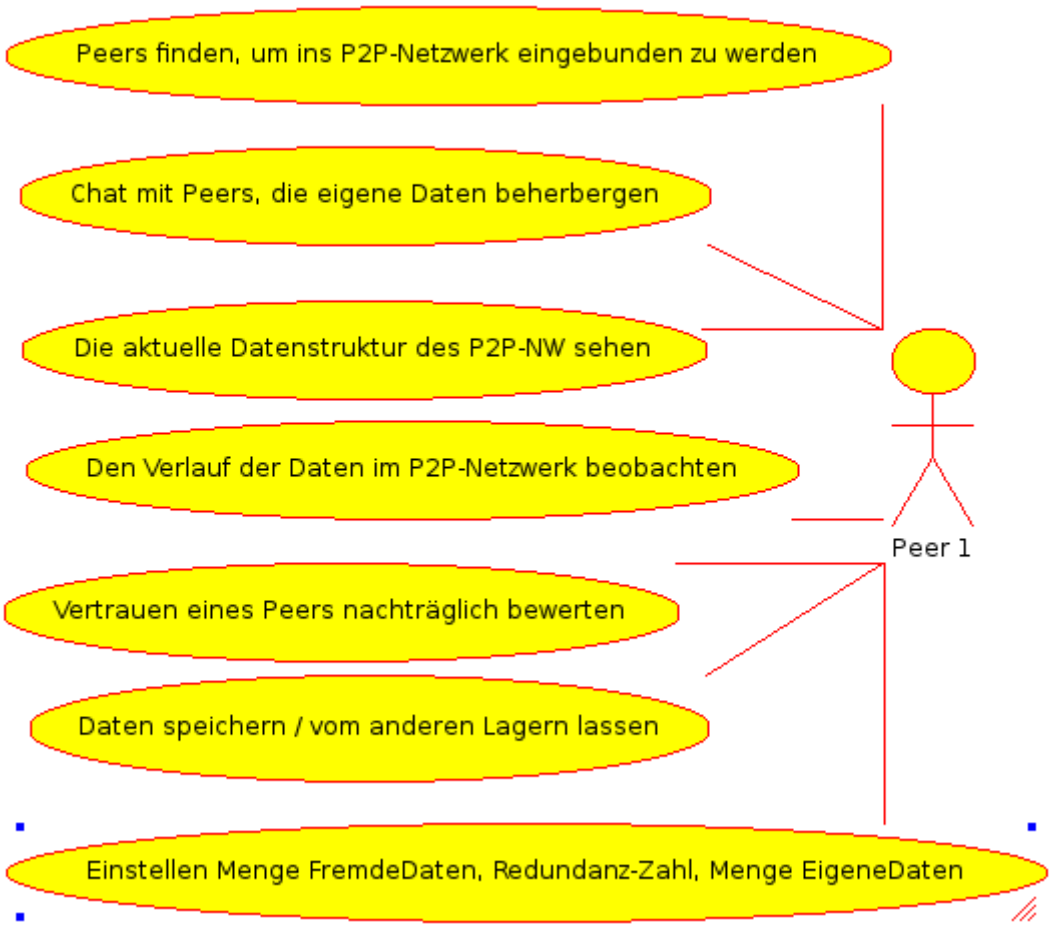


Abbildung 3.3 erhebt keinen Anspruch auf Vollständigkeit. Beim Erstellen von Prototypen und Klassendiagrammen werden weitere Problemkreise und Anwendungsfälle usw.

ersichtlich, die eingeplant werden müssen. Planungsgegenstände beeinflussen sich gegenseitig.

3.4. Hintergrundarbeiten eines Peers

Im Folgenden werden Programmfunktionen beschrieben, die dem Backend entsprechen, da im vorangegangenem Abschnitt Anwendungsfälle behandelt wurden. Die folgenden Programmaktionen geschehen teils abhängig und teils unabhängig von Interaktionen des Benutzers mit dem Programminterface oder indirekt über andere Programme, wie dem Dateimanager.

- Erstellung der Datenstruktur aus der das Netzwerk besteht
- Einigung mit anderen Peers mittels Ping-Broadcasts zu einem Zeitpunkt das Netz zu defragmentieren
- Registrieren mit File-Hooks, wann neue Daten im Ordner verändert wurden sind, damit sie mit der Peer-Cloud synchron sind.
- Behandeln von Exceptions, z.B. wenn ein Sync sich mit einem anderen Überschnitten hat, d.h. es gibt 2 Versionen einer Datei.
- Bewertungen eines Peers auf einer Menge von anderen Peers speichern, da es dafür keinen Server gibt
- Suchanfragen weiterleiten über die Hierarchie des Skipnets (von oben nach unten)
- Regelmäßige Überprüfungen, dass niemand zu viel bei anderen lagert und gleichzeitig genug Daten bei sich lagern lässt. Dafür muss in der Konfiguration ein Redundanzmultiplikator als Zahl festgelegt sein, der eine feste Untergrenze hat.
- Algorithmen zur wechselnden Einordnung eines Peers in eine Hierarchieeben. Es kann passieren, dass sich die Situation im Skipnet ändert und der betrachtete Peer höher oder niedriger im Skipnet einzuordnen ist.
- Verschlüsselung jeder Daten auf anderen Peers mit dem Hashwert des Passwortes des betrachteten Peers.

- Als späteres auswählbares Feature, die Zerstückelung von Datenmengen in Teile, damit sie eingeteilt auf einem fremden Peer nicht aus dem Passwort hergestellt werden können, sondern Daten aus anderen Peers nötig sind; dies ist ein Sicherheitsfeature.
- Auswahl eines Verfahren, um Datenverlust zu kompensieren. Dazu gibt es eine Zahl an Fehler korrigierenden Verfahren und Verfahren die Daten auf unterschiedliche Art vervielfachen z.B. Raid-Varianten. Raid 5 erlaubt beispielsweise den Verlust einer Festplatte beim Einsatz von 3 Festplatten. Raid 5 arbeitet mit Redundanz. Fehler korrigierende mathematische Verfahren können auch mit Redundanz arbeiten, als auch mit Datenabschnitten, die verwendet werden, um Teile eines Datenabschnittes, ggf. bei Verlust, wiederherzustellen. Es ist dann möglich auszusagen, wie viel Prozent eines Datenabschnittes wiederhergestellt werden kann und ab wann das nicht mehr möglich ist. Da Fehler korrigierende Verfahren nicht den Ausfall von dem Großteil an Datencontainern (Peers / Festplatten) kompensieren, eignen sich Methoden der Vervielfachung, wie bei RAID-Verfahren (z.B. RAID 1) besser für die P2P-Cloud.
- Aufrechterhalten und Schließen von Verbindungen zum Skipgraphen, mit “Keepalives”. Die Anzahlen lassen sich konfigurieren.
- Bei Passwortänderung müssen alle Daten neu verschlüsselt werden auf allen zugehörigen Peers.

Diese Punkte über Programmaktionen alle oben erheben nicht den Anspruch auf Vollständigkeit. Weitere Planungsinstrumente (z.B. andere UML-Diagramme) ermöglichen das weitere Auffinden von Teilproblemstellungen.

3.5. Architektur

Weil die Möglichkeit mehrere Userinterfacearten zu realisieren Vorteile birgt, Realisierungen der Software in mehreren Betriebssystemen einfacher möglich sind, und damit so genannte Unit-Tests durchgeführt werden können, ist die Model-View-Controller- oder Model-View-Presenter-Architektur geeignet. Im Folgenden wird das erklärt.

Model-View-Controller und Model-View-Presenter sind 2 Entwurfsmuster und Architekturen von Programmen auf Quelltextebene, die eine Unterteilung in 3 Bereiche von Klassen vornimmt. Es gibt Klassen die nur dem Backend angehören, Klassen die nur dem Frontend angehören. Dazwischen existieren Klassen, die als Vermittler zwischen beiden Klassen verwendet werden, in der Befehle quasi weitergereicht werden. So können einzelne Methoden und Quelltextabschnitte besser getestet werden, als wenn sich Backend und Frontend in den Klassen im Code überlappen. Dies geschieht mit so genannten Test-Klassen die extra dafür separat angelegt werden, denn so kann das Frontend umgangen werden beim Verwenden der Backendfunktionen in den Testklassen. Das sind Unit-Tests. Sie testen Komponenten, Funktionen, Module einzeln mit Testprozeduren, die erfolgreich oder unerfolgreich mit Return zurückgeben. Einige Entwicklungsumgebungsplugins, wie JUnit für Eclipse für Java unterstützen das Unit-Testen. Damit kann z.B. der Grad der vollständigen Abdeckung der Unit-Tests bestimmt werden.

Moderne Datenclouds verwenden Kernel-Filehooks. Diese benachrichtigen die Cloudsoftware darüber, dass und was sich im Dateisystem geändert hat. Somit muss nicht mit in regelmäßgen Abständen in Ordnern nach Veränderungen gescannt werden, was die Computer-Leistungs-Ressourcen unnötig beanspruchen würde. Zur Anwendung von Kernel-Filehooks eignet sich das Beobachter-Entwurfs-Muster als Architekturgrundlage. Das Beobachtermuster besteht aus der Subjektklasse, die beobachtet wird. In dieser können sich beobachtende Klassen registrieren, des Klassentyps “Beobachter”. Es sind also mehrere Beobachter pro Subjekt möglich. Bei Aufruf der Benachrichtigens-Methode im Subjekt werden alle Beobachter informiert, mit deren Methode “aktualisiere”. Andererseits ist es auch möglich, dass anstelle einer automatischen Benachrichtigung der Beobachter beim Subjekt nach dem Zustand quasi nachtragt mit der Methode des Subjektes “gib Zustand”. Das Subjekt verweist auf das Interface Beobachter und die Klasse Beobachter, die dieses Interface implementiert verweist auf die Klasse Subjekt.

Es macht Sinn das Backend in so genannte Pakete einzuteilen, die Funktionalität-Kategorien entsprechen. In den Paketen sind Klassen. Solche Kategorien sind beispielsweise Ver-

schlüsselung, RAID-angelehnte Technologien, Netzwerkkommunikationsprotokoll des P2P-Netzes, Erzeugung der Datenstruktur der Netztopologie mit anderen Peers, etc.

3.6. Suchstrategie

In Skipnetzen sind Bereichsanfragen möglich. Es macht Sinn, die eigenen entfernten Daten innerhalb so einer möglichen Bereichsanfrage lagern zu lassen. Dementsprechend müssen sich die Peers anordnen. Allerdings ist eine Bereichsanfrage ausgehend von jedem Peer wünschenswert. Von daher macht es Sinn, dass jeder Peer die Daten bei einem möglichst nahen Peer speichert. Jede Abweichung von dieser Herangehensweise verlängert die zukünftige Suchdauer.

Es ist wünschenswert, dass alle Peers lange online sind, denn so lässt sich die Position der Peers im Overlaynetzwerk besser aufrechterhalten, denn wenn Peers aus dem Netzwerk kurzfristig verschwinden, dann gelten andere Gesetze zum Einordnen in das Netzwerk.
Es ist möglich mit einer Metrik den Grad der Unordnung festzustellen. Das heißt, dass Daten von Peers nicht bei den Nachbarn gelagert werden. Ab einem gewissen Grad der Unordnung kann automatisch entschieden werden, ob Daten umkopiert werden oder ob Peers anders angeordnet werden.
Welcher Grad der Unordnung zu entscheiden ist, muss experimentell mit einem P2P-Netzwerksimulator ermittelt werden. Die Alternative ist es auf Bereichsanfragen nur vereinzelt zuzugreifen oder darauf zu verzichten.

Ohne Aufräumarbeiten gibt es des Weiteren die Möglichkeit mit mehreren Overlaynetzwerken zu arbeiten innerhalb des gleichen P2P-Netzwerkes. Ein Kompromiss dagegen ist es, von oben nach unten ab einer gewissen Schicht weitere Teiloverlaynetzwerke zu etablieren. Die optimale Lösung muss experimentell ausgelotet werden, was bisher nicht möglich ist, weil noch keine Software programmiert wurde, mit der man experimentieren

könnte und das den Rahmen sprengen würde.

Es genügt nicht, wenn Peers oberer Schichten auf Peers unterer Schichten verweisen. Peers müssen zudem Informationen abrufbar machen, welche Daten sie besitzen, wenn Benutzername und Passwort für diese Daten angegeben wird. Denkbar ist auch, dass die untersten Peers im Skipnet Informationen besitzen, wo die Daten ansonsten liegen, wenn nicht bei Ihnen.

Die Möglichkeit besteht, dass ein eigener P2P-Netzwerksimulator auf Basis eines anderen geschrieben werden muss. Das bedeutet, dass vorhandener Quellcode oder vorhandene Netzwerksimulatoren genutzt werden, die sich nicht auf P2P-Netze beschränken, d.h. dass jede Netzwerksoftware damit simuliert werden kann, die mit der Simulationssoftware verwendet wird. Quellcode kann erweitert werden und Netzwerksimulatoren können als Programmbibliothek oder Programm genutzt werden.

3.7. Gestaltungsrichtlinien der P2P-Cloud

Der Fokus des P2P-Netzes liegt darauf die Funktionalitäten für den Endanwender unsichtbar zu machen, dass wenig oder nichts konfiguriert werden muss, damit das Produkt funktioniert. Es ist von Vorteil, wenn das Frontend übersichtlich ist, und es eine Begrenzung der Einstellmöglichkeiten gibt. Eine Ausnahmebehandlung über Programmfehler oder unerwünschte Programmzustände (z.B. Netzwerkabbruch), auch als Exception-Handling bekannt, muss so weit es geht automatisch erfolgen.

4. Schluss

4.1. Ergebnisse und Untersuchungen

Untersucht wurde in dieser Arbeit hauptsächlich welche Netzwerkdatenstruktur / Topologie sich für eine P2P-Cloud eignet. Weiterhin wurde untersucht wie man Datenverlust entgegnen kann, der bei einer P2P-Daten-Cloud prinzipiell eher vorliegen kann, als in einer Server-Client-Architektur. Das hat den Grund darin, dass der jeweilige Server in der Regel einer Firma gehört, die sich Vertrauen sichern muss, um wirtschaftlich zu bleiben. Dem Datenverlust entgegnet im P2P-Netz hierbei

- ein Chat der sozialen Bindungen dienen soll, mit dem Peers, die gegenseitig Daten lagern Kontakt halten können;
- die Beobachtungsmöglichkeit wo die eignen Daten entfernt lagern,
- die automatische und manuelle Bewertungsfunktion,
- die verstellbare Redundanz-Zahl.

Da ein Skipnet mehrere Ebenen hat, eignen sich diese Ebenen der entsprechenden Zuweisung von Hierarchiestufen, die Peers zugewiesen werden. Dazu kommt, dass sorgfältig berechnet werden muss, welcher Peer in welche Ebene eingeklinkt wird, nach den Kriterien Datendurchsatz und Ressourcen.

Die Recherche hat ergeben, dass es dutzende P2P-Netzwerk-Simulations-Produkte verschiedener Art (bzgl. Netzwerklayer) gibt. Darauf kann zurückgegriffen werden, wenn schon etwas funktionierendes implementiert wurde.

Es kann Sinn machen, wenn 2 Peers die Daten von anderen Peers besitzen, diese austauschen (3.6. Suchstrategie). In der Untersuchung hat sich herausgestellt, dass dafür der Grund vorliegt, wenn durch das Umkopieren langfristig für beide Seiten nachträglich ein Leistungsgewinn entsteht (Latenzzeiten und Datendurchsatz). Ein anderer Grund liegt vor, wenn beide Seiten durch den Datenaustausch bzw. -Wechsel eine höhere Verfügbarkeit der fremden Daten in ihren Zeitrahmen ermöglichen, wenn die durchschnittlichen Onlinezeiten am Tag beider durch eine Software untersucht wird. Wenn alle entfernten Peers, die eigene Daten lagern, zu möglichst unterschiedlichen Zeiten online sind, ist eine höhere Verfügbarkeit gegeben, als wenn sie zu gleichen Zeiten online sind.

4.2. wichtige Ergebnisse und Aussagen

Diese Arbeit dreht sich zum größten Teil um die Wahl der Topologie für die zu spezifizierende P2P-Cloud. Das Ergebnis davon ist die Wahl des Skipgraphen, weil die Bereichsanfragen die damit möglich sind sich nützlich erweisen alle entfernt redundant gelagerten Daten eines Peers sofort auffindbar zu machen. Das Auffinden eines Peers im Skipnet beträgt dabei O(log n), wenn auch Hashlisten O(1) bieten würden. Wenn In einem P2P-Skipnet willkürlich sich mehrere Peers trennen ohne einem Protokoll zu folgen, also z.B. durch Internetstörungen, dann liegt eine teilweise Zerstörung des Skipgraphen vor. Der zweite Hauptgrund zur Entscheidung für das Skipnet ist der Grad der Reparierbarkeit solcher teilweise zerstörter Netzstrukturen.

Eine wichtige Aussage dieser Arbeit ist, dass es in einer P2P-Daten-Cloud immer das Risiko gibt, dass Datenverlust auftritt, wenn entfernte Peers mit den eigenen Daten z.B. dauerhaft offline geht.

Es hat sich herauskristallisiert welche weiterführenden Fragen und Teilgebiete zu klären und zu vertiefen sind in weiteren Schriften zur P2P-Cloud, z.B. die konzeptionelle Sicherheit eines P2P-Netzes im Bezug zu inoffiziellen Quellcode-Forks eines Peers, die auf einmal anders agieren mit den ursprünglichen Peers. Wie Routing vonstatten geht muss geklärt werden und wie die Peers kooperativ ohne zentralen Server die Skipgraph-

struktur gemeinsam herstellen.

Unverschlüsselt Daten lagern auf einem beliebigen entfernten Peer ist nicht zumutbar. Auch wenn ein Nutzer verschlüsseln kann mit Produkten von Dritten ist es angebracht Daten in der Cloud automatisch zu verschlüsseln mit einem Modul der P2P-Cloud.

4.3. Ausblick und Alternativen

Die Alternativen zu einer P2P-Cloud sind u.a. vorhandene Server-Client basierte Cloudlösungen, von denen es dutzende gibt, mit sich unterscheidenden Geschäftsmodellen und auch als Opensource (OwnCloud). Die einfachste Alternative ist das manuelle kopieren mit einem Dateimanager. Backup- und Synchronisationsprodukte seien hier auch genannt.

Die Firma Symform hat jedoch schon eine P2P-Cloud entwickelt. Man kann dort mehr Speicherplatz in der Cloud erwerben, wenn man mit eigenem Speicherplatz oder Geld “bezahlt”, siehe http://www.symform.com ![Garg(2012)] Für die Bereitstellung von 2 GB, kann man 1 GB “bekommen”, also liegt eine 2-fache Redundanz vor. Die P2P-Cloud dieser Arbeit sieht eine 10-fache Redundanz vor, bzw. eine die regelbar ist. Beides hat offensichtliche Vor- und Nachteile. Im Internetforum unter [viele(2012)] finden sich 46 Erfahrungsberichte und Meinungen in Abständen von Monaten mit der Software von Symform. Insgesamt wird dort die Idee befürwortet, die Umsetzung kritisiert und bemerkt, dass besser nur unwichtige Daten damit gesichert werden sollten, wegen dem Verlustrisiko. Verschlüsselung gibt es sowohl in der Symform-Lösung als auch Fragmentierungstechnologie, wie bei RAID 0. Vorteil der P2P-Cloud dieser Arbeit ist jedoch der frei einstellbare Redundanzfaktor, sind die Bewertungsfunktionen, die in der Arbeit genannten Vorteile des Skipnets; all das um Zuverlässigkeit zu gewährleisten.

Mit dieser Arbeit konnte die Problematik nur systematisch angerissen werden, wozu ein Überblick der essenziellen Fakten vorhandener Technologie in Kapitel 2.2 erstellt wurde, um sich darin zu bedienen. Es würde den Rahmen sprengen, genauer auf Sicherheits-

fragen einzugehen, auf die Problematik der Herstellung des Skipnetzes durch Peers.

Es ist davon auszugehen, dass P2P-Clouds in der Zukunft weiter perfektioniert werden und neue Lösungen entstehen, trotz Software-Patent von Symform, weniger Bugs enthalten, ressourcenschonender werden (Verschlüsselung kostet z.B. Ressourcen) und mehr Technologien enthalten werden, um dem Problem der Zuverlässigkeit bezüglich Datenverlust zu entgegnen.

A. Abbildungsverzeichnis

B. Literaturverzeichnis

[Dmitry Korzun(2013)] Dmitry Korzun, A. G., - 2013. Structured peer-to-peer systems [elektronische ressource] : Fundamentals of hierarchical organization, routing, scaling, and security. ebook, iSBN: 978-1-461-45483-0 EAN: 9781461454830.
URL http://link.springer.com/book/10.1007/978-1-4614-5483-0/page/1

[Garg(2012)] Garg, P., 4 2012. Can symform's p2p cloud storage platform really work?
URL http://www.forbes.com/sites/danwoods/2012/12/04/can-symforms-p2p-cloud-storage-platform-really-work/

[Hildebrandt(2007)] Hildebrandt, D., 2007. Peer-to-Peer Systeme simulationsbasiert entwickeln mit RealPeer. Verlag Dr. Müller, iSBN-13: 978-3836445122 ISBN-10: 3836445123.

[Mahlmann(2007)] Mahlmann, P., 2007. Peer-to-peer-netzwerke [elektronische ressource] : Algorithmen und methoden. ISBN: 978-3-540-33992-2.
URL http://www.springerlink.com/content/u6kl3uhttp://dx.doi.org/10.1007/978-3-540-33992-2undArchivserverderDeutschenNationalbibliothek

[viele(2012)] viele, 2012. Symform cloud-backup im paketzentrum erfahrungen?
URL http://www.synology-forum.de/showthread.html?32992-SymForm-Cloud-Backup-im-Paketzentrum-Erfahrungen