Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Lutz Behnke

Ad-Hoc Distributierter Speicher

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Lutz Behnke

Ad-Hoc Distributierter Speicher

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Master Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Martin Hübner

Abgegeben am 11. Oktober 2006

Lutz Behnke

Thema der Masterarbeit

Ad-Hoc Distributierter Speicher

Stichworte

Netzwerk Persistenz Storage

Kurzzusammenfassung

Wotan ist ein Systemvorschlag, um die Notwendigkeit der manuellen Konfiguration von virtuellen Laufwerken und der Organisation von Ablagen durch den Benutzer zu eliminieren. Software soll die Aufgabe der Beschaffung von Speicherresourcen und die Wiederbeschaffung von Dateien erledigen. Dabei kooperieren mehrere Computer um Daten dauerhaft zu speichern (für den Benutzer transparent) und an jedem Ort wieder verfügbar zu machen. Dies hat zum Ziel, den Ablageort der Daten von einer bestimmten Maschine zu lösen und an Benutzer und Prozesse zu binden. Kooperative Speicherung soll helfen, den notwendigen Konfigurationsaufwand zu reduzieren und für den Benutzer wahrnehmbar, Stablität und Robustheit gegen Fehlerfälle zu verbessern, selbst dann, wenn einzelne Maschinen im Verbund ausfallen oder permanent zerstört werden.

Lutz Behnke

Title of the paper

Ad-Hoc Distributed Storage

Keywords

Network Persistence Storage

Abstract

Wotan is a proposal for a system to eliminate the necessity to manually configure virtual drives and the organisation of filing systems by the user. Software should handle the task of procuring storage rescources and the retrieval of data. To achieve this, a number of computers cooperate to store the data persistently and deliver it to any place needed. The goal is to separate the storage place from a particular machine and bind it to a user or a process. Cooperative storage is tasked to aid in reducing the configuration effort and improving stability and robustness against failure, as perceived by the user. Even in the face of machine failure or even permanent destruction.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Das Problem

Eine wachsende Gruppe von IT Bereichen beschäftigt sich mit der Verwendung einer Vielzahl von Computern, um dem Anwender eine verteilte Dienstleistung zur Verfügung zu stellen. Dabei werden in der Regel auch Dateien erzeugt oder verarbeitet. Die Anzahl der beteiligten Knoten (siehe Abschnitt A) übersteigt dabei die Anzahl der Benutzer in ständig steigenden Größenordnungen. Und auch die Anzahl der Speicherobjekte ist nicht mehr durch den Anwender zu überblicken[1]. Im Bereich des *Ubiquitous Computing* (**?**,**?**, **?**) ist jeder Anwender von allgegenwärtigen und autonom agierenden Geräten umgeben und wäre mit einer manuellen Konfiguration dieser Knoten überfordert. Dieser Effekt wird besonders dadurch verstärkt, dass Anwender die Computer nicht als solche identifizieren können, da Objekte des täglichen Lebens[2] mit Logik ausgestattet sind und u.U. Daten ablegen möchten.

Auch im Bereich des *Grid Computing* (**?**, **?**, **?**), mit domänen-übergreifenden Anwendungen, und im Bereich des *Cluster Computing*, mit seiner enormen Anzahl an Rechner-Knoten, sind Verbesserungen bei Konfigurierbarkeit und nutzbarer Stabilität notwendig. Untersuchungen (**?**) zeigen, dass ein Großteil der Komplexität von existierenden Grid-Systemen, wie dem Globus Toolkit (**?**), der Überwindung von Organisationsgrenzen dient, und damit eigentlich wenig mehr sind als ein FTP (**?**) mit verbesserter Zugriffsrechte-Verwaltung.

Im einzelnen können in den genannten Bereichen folgende Probleme identifiziert werden:

[1] Bsp.:Im Heimatbereich meines Hochschulkontos befinden sich 159725 Dateien. Ich kenne nur einen Bruchteil davon.

[2] der sprichwörtliche Toaster

1.1.1 Konfiguration

Kein Mensch, egal ob Fachmann oder Laie, kann die sich ständig verändernde Zusammenstellung von Geräten und Kapazitäten in den obigen Bereichen zeitnah konfigurieren. Daher benötigen alle Elemente die Fähigkeit, sich selbständig nach gewissen Regeln in ihrer sich u.U. ändernden Umgebung zu konfigurieren.

Neben dem Erfordernis eine arbiträre Anzahl von Elementen unterstützen zu müssen[3], stellt vor allem die sich ständig ändernde Umgebung und Netzzusammenstellung eine hohe Anforderung an ein Speichersystem. Bei den begrenzten Reichweiten von Bluetooth und WLAN muss ein System trotz ständig erscheinenden und verschwindenen Knoten stabil die Datenübertragung fortsetzen.

Projekt Oxigen(**?**) beschreibt einige der Anforderungen aus Sicht des Ubiquitous Computing.

1.1.2 Usability

Neben dem technischen Problem der generellen Konfigurierbarkeit von Geräten wächst mit der Anzahl der Speicherobjekte auch die Schwierigkeit für Menschen, diese Objekte zu identifizieren und wiederzufinden. Vor allem werden für Laien unverständliche technische Details in dem Namen eines solchen Objektes wiedergespiegelt.

Wenn der Ablageort einer Datei seinen Namen bestimmt, so ist der Anwender durch die Ablagestruktur der Person, welche die Datei abgelegt hat gebunden, und kann nur sehr schwer eigene Notizen oder Ordnungsysteme benutzen. Besonders deutlich wird dies bei der Verwendung von URLs: (z.B `http://sample-server.domain.com/some/path/file.mpg`) Hier wird nicht nur der Ablagepunkt vorgegeben, definiert durch Server-Name und Pfad in der Speicherhierarchie, sondern auch noch das Protokol festgelegt, mit dem auf dieses Speicherobjekt zugegriffen werden muß.

In meiner Erfahrung ist den meisten Laien die interne Struktur von URLs nicht gegenwärtig und die Möglichkeit sich URLs zu merken, sinkt rapide mit der Länge des Namens. Was für einen Informatiker durch Kenntnis um den Aufbau der URLs noch erinnert werden kann, bleibt für den Laien nur ein unverständliches Durcheinander aus Buchstaben, Punkten und Schrägstrichen.

Die wachsende Verbreitung von Desktop-Such-Funktionen, welche die eigenen Dateisysteme durchsuchen, sind ein Beleg für die wachsende Dringlichkeit dieses Problems. Die meisten Benutzer haben wahrscheinlich längst aufgegeben sich zu merken wie und wo sie Dateien abgelegt haben, und suchen lieber nach ihnen. Doch die Desktopsuche ist ein Artefakt der Desktop-Metapher, bei der jedem Benutzer ein einzelner Computer zugeordnet

[3] Es gibt in Deutschland mehr als 80 Millionen Einwohner und im Juni 2005 74.1 Millionen Handys. Und das obwohl 30% aller Deutschen bei einer Umfrage sagten das sie gar kein Handy haben wollen

ist. In der Ubiquitous Computing Metapher gibt es keine klare Trennung zwischen *meinem* Computer und dem Rest des Internets. Alle intelligenten Knoten sind verbunden, und ihre Zusammensetzung ändert sich kontinuierlich.

1.1.3 Stabilität und Ausfallsicherheit

Mit einer steigenden Anzahl von Geräten die ein Anwender sein eigen nennt, steigt auch die Notwendigkeit festzustellen welche Daten auf welchem System liegen und ob sie irgendwo noch mal als Sicherheitskopie abgelegt sind.

Aber in der Realität (jedenfalls in meinem erweiterten Freundeskreis, der sowohl Computer-Experten als auch absolute Computer-Hasser umfasst) haben die meisten Anwender (vor allem solche ohne besondere Computerkenntnisse) überhaupt keine Sicherheitskopien. Backup-Systeme auf Band, zusätzlichen Festplatten oder in verwalteten CDs, sind immer noch teuer, oft nur von Fachleuten zu konfigurieren, und ein Speichern jeder geänderten Datei auf CDs oder DVDs sorgt nur für einen wachsenden Stapel von Medien, die nie wieder angefasst werden.

Vor allem aber muß ein Mensch bei dem Verlust der Inhalte eines Speichermediums mühsam die Berge von anderen Medien durchwühlen um herauszufinden welche denn die aktuelle Kopie einer Datei ist. Systeme die so etwas mit Hilfe von Datenbanken und zentralen Repositories leisten, sind eindeutig nicht im Bereich des Heimanwenders zu finden.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Erarbeitung eines Konzeptes zur Verwaltung von Speicherobjekten. Dies beinhaltet den Entwurf einer Middleware, welche für den Anwender vollkommen transparent ist. Daneben umfasst das Konzept auch eine Umgebung, in der die Speicher-Geräte zur Abrechnung von Diensterbringung und der Übernahme von Risiko eingebettet sind.

Dieses Risiko entsteht durch die Übernahme der Daten eines Benutzers durch einen anderen. Wenn der Erzeuger der Daten diesen einen gewissen Wert beimisst, so muß ein Verlust der Daten bestraft oder zumindest als Wertverlust abgerechnet werden. Die Arbeit beschäftigt sich weder mit dem exakten Berechnungs-Modell der Wertschöpfung durch die Diensterbringung und der Abrechnung des Wertes von Risikoübernahme, noch der Berechnung von Zahlungen bei Datenverlust. Es liefert lediglich die notwendige Infrastruktur, um eine solche Abrechnung durchzuführen.

Für den Benutzer sollte die gesamte Arbeit der Middleware transparent sein. Er muß nur jene Geräte “markieren”, die ihm gehören. Er hat dann zu jeder Zeit auf jedem dieser Geräte den Zugriff auf alle seine Dateien, ohne das er sich Objekt-Namen oder Speicherorte merken muß. Außerdem muß der Anwender keine Konfiguration von Geräten durchführen.

Die Infrastruktur muß auch auf Geräten in Mobile Ad-Hoc Networks (MANets) wenigstens ein Anlegen von Dateien erlauben. Ein lesender Zugriff des Benutzers auf alle seine Daten kann sich durch die begrenzte Übertragungskapazität u.U. verbieten.

Außerdem müssen alle Daten gegen einen unberechtigten Zugriff geschützt sein. Dabei muß die Frage der Berechtigung direkt vom Anwender und nicht durch einen zentralen Administrator geregelt werden können.

1.3 Aubau der Arbeit

Zur Erstellung eines tragfähigken Entwurfes werden in Kapitel 2 detailliert die technischen und organisatorischen Anforderungen an ein solches System untersucht.

Kapitel 3 gibt einen kurzen Überblick über die eingesetzten Technologien, Kapitel 4 stellt alternative System-Designs vor und stellt sie den Anforderungen gegenüber.

Kapitel 5 stellt den Entwurf eines Systemes vor, das versucht diese Anforderungen zu erfüllen. Die einzelnen Komponenten sowie die eingesetzten Protokolle werden beschrieben.

Der Arbeit liegt kein durchgängig einsetzbarer Prototyp zugrunde, sondern es wurden nur einzelne Elemente durch kleine Proof-of-Concept Programme untersucht.

Ein Teil der notwendigen Protokolle für die unterschiedlichen Aufgaben des Systemes sind ausreichend genau definiert, um eine Implementation zu beginnen. Diese sind im Anhang angefügt.

Kapitel 2

Anforderungen

Um die Anforderungen an ein System wie Wotan besser zu verstehen, ist im folgenden Abschnitt ein typisches Anwendungsszenario beschrieben. Es ist nicht das einzige Anwendungsfeld für kooperative Speichersysteme, aber es ist für die Entwicklung von Wotan als leitende Anwendung benutzt worden. Andere Anwendungszenarios, wie ein Netzwerk-weites Backupsystem oder Softwareverteilung können zu anderen Lösungen (z.B. PAST, Abschnitt 4.3.5.1, CFS, Abschnitt 4.3.4.2 respektive) führen.

In den nachfolgenden Abschnitten werden die aus dem Szenario resultierenden Anforderungen detailiert definiert.

2.1 Die Dschungel-Patrouille

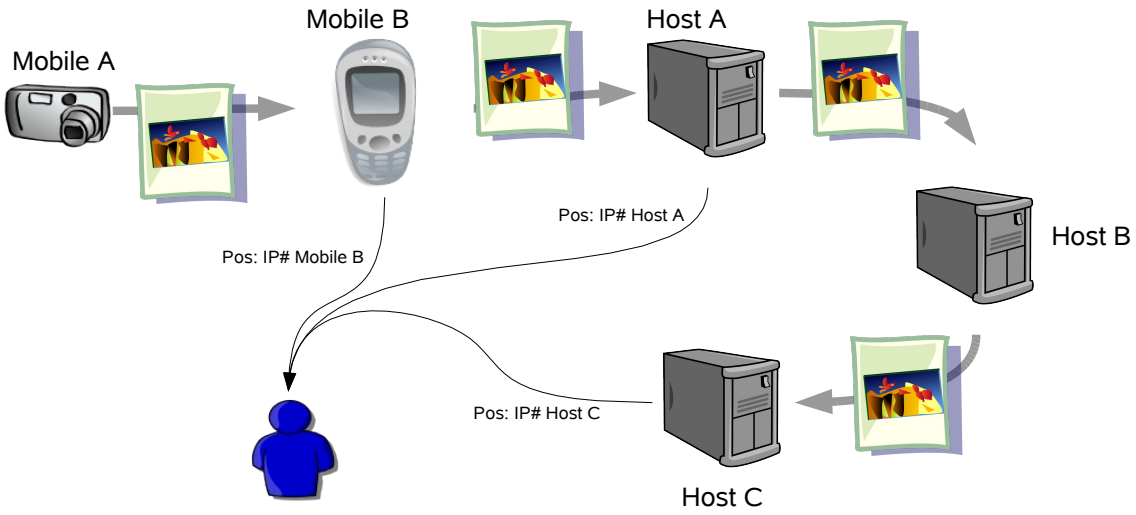

Abbildung 2.1: Anwendungszenario für Wotan

Ein Ferienklub in tropischen Gefilden organsiert einen Ausflug in den nahegelegenen Regenwald. Man hat zwar einen Führer, und die GPS-Lokalisation funktioniert dank der Satelliten auch, aber durch dichten Bewuchs und einen Bergrücken ist man von jeder sonstigen Funk-Kommunikation abgeschnitten. Die Teilnehmer der Reisegruppe aber sind moderne Menschen und tragen eine Unzahl an mobilen IT-Geräten mit sich herum: Digitalkameras, Mobiltelephone, PDAs, XDAs[1]. Natürlich sind auch genügend Ersatzbatterien dabei, um nicht in Energie-Not zu kommen. Einige der Teilnehmer photographieren, was das Zeug hält, andere lassen lieber die Umgebung direkt auf sich wirken. Dies führt natürlich dazu, daß einige Geräte schneller ihre Speicherkapazität ausgeschöpft haben als andere.

Um nun aber auf keinen Fall ein Bild zu verlieren gibt es einen Softwaredienst, der dafür sorgt, das die Geräte mit den vollen Speichern bei den Geräten mit den leeren Speichern anfragen, ob es nicht möglich wäre, einige Dateien bei ihnen auszulagern. Und die Geräte mit freien Ressourcen entscheiden, dass dies möglich sei und übernehmen die Bilder. Davon merken die Benutzer natürlich nichts, sondern das handeln die Geräte ebenso selbstständig aus wie die Frage, wer mit wem in einem Ad-Hoc Netzwerk überhaupt kommunizieren kann.

Sind nun alle erschöpft, gehen sie nach Hause. Und sobald die mobilen Geräte bemerken, dass sie an einem Internet-Hot-Spot vorbeikommen, versuchen sie sich der Verantwortung zu entledigen (denn so ein Handy kann ja leicht mal geklaut werden oder einfach zu Boden fallen und kaputt sein.). Daten und Verantwortung werden vom Hot-Spot natürlich sofort an einen anderen Computer im Internet weitergeleitet, der, eingebaut hinter dicken Mauern und sowohl mit Klimananlage als auch mit Backup-Tapes ausgestattet, die Verantwortung gerne übernimmt.

Nach dem Duschen und dem Abendessen kann sich der Teilnehmer der Dschungel-Patrouille an seinen Laptop setzen und auf alle Bilder zugreifen, egal ob er sie mit dem eigenen PDA aus dem Dschungel getragen hat, oder ob jemand diese Aufgabe für ihn übernommen hat, ohne es zu merken.

2.2 Anforderungen im Detail

2.2.1 Discovery

Jede im System beteiligte Komponente muss die von ihr benötigten anderen Komponenten selbstständig finden können oder in einen geeigneten Zustand übergehen, wenn dies nicht möglich ist. Auch die Parameter, die für einen ordnungsgemäßen Betrieb notwendig sind, müssen entweder für eine möglichst große Gruppe von Computern einheitlich sein oder durch geeignete Protokolle ausgehandelt werden.

[1] Geräteklasse zwischen den mobilen Telephonen und richtigen PDAs. Meist mit Ressourcen ausgestattet, die an einen günstigen PDA heranreichen, aber in einem Telephon-Formfaktor.

Der kleinste gemeinsame Nenner für die Computer im Wotan -System sind “embedded” Computer ohne Benutzerschnittstelle, oft können sie vom Benutzer nicht einmal als Computer erkannt werden. Daher dürfen evtl. existente Konfigurations-Parameter auch nur extern zu den beteiligten Systemen und einheitlich für alle diese Systeme gesetzt werden.

2.2.2 Read-Write Operationen

Viele Systeme auf Basis von Peer-2-Peer Protokollen wie das Cooperative Filesystem (CFS, **?**, siehe Abschnitt 4.3.4.2) bieten nur lesende Operationen. Das Schreiben von Daten in das Speichersystem geschieht in einem Out-Of-Band (OOB) Verfahren, das nicht Bestandteil des Speichersystems ist.

Das Anlegen von Speicherobjekten, auch auf Systemen ohne Benutzerschnittstelle, ist eine wesentliche Anforderung von Wotan .

Das Ändern von einmal angelegten Objekten ist keine Anforderung. Hier kann bei Bedarf stattdessen die Magnet-Band-Metapher einer Turingmaschine verwendet werden: Die Daten eines Speicherobjektes werden ausgelesen, verarbeitet und in eine neue Datei geschrieben.

2.2.2.1 Freshness Garantie

Wenn das System Operationen für eine Änderung von Datei-Inhalten zur Verfügung stellt, dann muß es auch garantieren, dass diese Änderungen an alle Knoten propagiert werden oder ein Zugriff auf geänderte Objekte verhindert wird.

Sowohl Änderungen an Dateien als auch auch die Änderung von evtl. vorhandenen Meta-Informationen müssen von einem Knoten auch umgehend bei allen anderen Knoten sichtbar sein und vor allem, muss sichergestellt werden dass andere Knoten den Versuch einer Manipulation durch einen Man-in-the-middle Angriff zumindest detektieren können.

Kann ein System keine Aktualität gewährleisten, so muß dies zumindest durch den Knoten detektierbar sein, entweder durch geeignete Betriebszustände oder durch vom Design vorgegebene Eigenarten des Systems.

Kann das System die Aktualität von Daten nicht sicherstellen, so darf es die entsprechenden Primitiven (Ändern von Dateien, Ändern von Metadaten) nicht anbieten.

Gründe für diese Forderungen und die Möglichkeiten für einen Missbrauch werden in (**?**) untersucht.

2.2.3 Ende zu Ende Verschlüsselung

Sicherheit wird von Benutzern meist nicht als wichtig eingeschätzt, wenn sie dafür zusätzlichen Aufwand in Kauf nehmen müssen. Daher muß jede Sicherheitsmaßnahme ständig und

ohne Benutzerinteraktion durchgeführt werden. Aus diesem Grund sollten immer alle Nutzdaten in Wotan verschlüsselt werden, und die notwendigen Schlüssel ebenso transparent verwaltet werden wie die Nutzdaten auch.

Vor einem Austausch von Schlüsseln zwischen zwei Computern ist es darüber hinaus notwendig, das beide Computer gegenseitig ihre Identität durch den Einsatz von starker Authentisierung überprüfen.

2.2.4 Skalierbarkeit

Die Gesamtzahl der an das Internet angeschlossenen Geräte ergibt potentiell die Zahl der Anwender des Speichersystems. Daher darf es keinen einzelnen Punkt im gesamten System geben, der nicht potentiell mit der Anzahl der Clients mitwachsen könnte.

2.2.5 Stabilität gegen Datenverlust

Da eine wichtige Gruppe von Zielplattformen für Wotan billige PDAs und andere mobile Geräte sind, die schon durch ihre Benutzung außerhalb einer sicheren Büroumgebung einem höheren Risiko durch Ausfall, Beschädigung oder Zerstörung ausgesetzt sind, muss das System auch mit dem katastrophalen Versagen jeder einzelnen Komponente zurechtkommen, ohne Daten zu verlieren.

2.2.6 Stabilität bei Netzwerk-Partionierung

Bei mobilen Umgebungen kann davon ausgegangen werden, dass Konnektivität zum Internet nicht ständig oder nicht in ausreichendem Umfang oder mit ausreichender Verbindungs-Stabilität zur Verfügung steht. Das Beispiel-Szenario in Abschnitt 2.1 macht diesen Umstand sogar zu einem wesentlichen Bestandteil der dargestellten Umgebung.

Innerhalb des Systemes darf auch eine Netzwerk-Partitionierung nicht zu Datenverlust führen.

2.2.7 Abrechnung von erbrachter Leistung

Unter der Prämisse, dass die Elemente des Systems unterschiedlichen Personen oder organisatorischen Einheiten gehören, ist es nicht wahrscheinlich, dass in einem flächendeckenden Einsatz alle Teilnehmer freiwillig und ohne gefühlte Gegenleistung ihre Bandbreite und ihre Speicherkapazität zur Verfügung stellen. Damit erhalten beide Ressourcen einen realisierten Wert, und für ihre Verwendung muß der Eigentümer kompensiert werden.

Auf der anderen Seite stellen für den Erzeuger der Daten diese auch einen gewissen, wenn im Einzelfall auch geringen Wert dar. Z.B. bei mehreren hundert Urlaubsbildern ist das einzelne auch dem erstellenden Anwender nicht wirklich viel wert, aber die Gesamtheit stellt

schon einen beträchtlichen, wenn auch ideellen Wert dar. Wäre dem nicht so, hätte sich der Erzeuger nicht die Mühe gemacht, die Aufnahmen zu erstellen.

Die Abrechnung der Leistung muß sowohl die Übernahme des Risikos für den Wert als auch die erbrachte Weiterleitung oder dauerhafte Speicherung belohnen, als auch bei Verlust der Daten eine Strafe verhängen. Ob diese Wertschöpfung jemals mit realem Geld abgerechnet wird, hängt von dem endgültigen Einsatz-Szenario des Systems ab.

2.2.8 Minimale Asymetrische Kryptographie

Die Energieeffizienz ist in Wotan ein relevanter Faktor. Unabhängig davon, wie weit die Entwicklung von mobilen Plattformen voranschreitet, ist bei diesem Formfaktor die verfügbare Energie immer eine beschränkende Ressource. Daher können auch die Rechenleistungen nicht im gleichen Maße wachsen wie bei ortsgebundenen Maschinen.

Asymetrische Kryptoverfahren sind in der Regel ca. 1000 mal langsamer als symetrische Verfahren von gleicher Stärke. Daher sollten asymetrische Verfahren nur dann eingesetzt werden, wenn es keine alternative Möglichkeit gibt, die notwendige Funktion mit anderen kryptographischen Primitiven abzubilden.

2.2.9 Sonstige Anforderungen

Übliche Anforderungen, wie niedrige Latenz für Schreib- und Lesezugriffe oder auch die einzelne ebenso wie die aggregierte Bandbreite müssen zur Erfüllung der anderen Anforderungen in den Hintergrund treten. Sollten sie natürlich derart anwachsen, das mit dem System nicht mehr praktisch gearbeitet werden kann, so kann ein Systementwurf ebenso als gescheitert gelten, als wenn die oberen Anforderungen nicht erfüllt wären.

Kapitel 3

Grundlagen

In Wotan werden einige Technologien verwendet, die (noch) nicht Alltag in der Informatik sind. Diese sind im Folgenden kurz beschrieben.

3.1 Distributed Hash Table (DHT)

Eine DHT stellt sich aus der Sicht ihrer API wie jede andere Hashtabelle auch dar: Sie verwaltet Paare aus Schlüssel und Wert. Einziger Unterschied der meisten Implementationen ist allerdings der Umstand, dass als Wert immer nur ein Speicherort, die IP-Adresse eines Computers, zurückgegeben wird. Beliebige Daten ablegen zu können ist dann die die Aufgabe von höheren Ebenen eines Anwendungssystemes.

Die Verteilung in der *Distributed* Hash Table findet nun durch eine Zuweisung der Einträge der Hash-Tabelle auf verschiedene Computer statt, so dass auf jedem Computer nur ein Teil der in der DHT gespeicherten Daten abgelegt ist. Je nach Implementation des DHT sind die Computer entweder in einem Ring oder in einer Struktur, die einem B-Baum ähnelt, organisiert. Auch beliebige andere Organisations-Strukturen sind denkbar, in der Forschung bisher aber nicht untersucht worden.

Wie in jeder anderen Hash-Tabelle auch muß der Schlüssel in einen definierten Zahlenraum abgebildet werden. Hierfür werden kryptographische Hashfunktionen wie MD5 oder SHA eingesetzt. Wird das Ergebnis als Integer-Zahl gewertet, lassen sich alle Schlüssel in einen Zahlenstrahl (bzw. einen Ring wie bei Chord) übertragen und mit Hilfe von Intervall-Aufteilung auf die Knoten verteilen.

Ein Protokoll, dass zwischen den Knoten durchgeführt wird, sorgt für eine Reorganisation der Tabelle und einen Austausch von Speicherorten, wenn Knoten die DHT betreten oder verlassen.

Die Details dieses Protokolls und die Geschwindigkeit, mit der die DHTs auf diese Änderungen reagieren, ist sehr stark von den gewählten Designs, aber auch von Einstellungs-Parametern des jeweiligen Designs abhängig. Diese Einstellungs-Parameter können die Ei-

genschaften eines DHTs über das grundlegende Design hinaus maßgeblich bestimmen. Optimiale Wert für die Parameter sind vor allem stark von der Anzahl und Geschwindigkeit der eingesetzten Computer abhängig. (**?**) werden unterschiedliche DHTs, mit Variationen ihrer Parameter, auf ihre Leistungsfähigkeit hin untersucht. Aufgrund dieser Untersuchungen wurde Chord (**?**) ausgewählt.

3.1.1 Overlay Netzwerk

Die Verwendung einer DHT realisiert eine Netzwerk-Struktur, welche das physikalische Netz überlagert. Diese Struktur ergibt sich aus dem Umstand, dass nur eine Untermenge der Internet-Knoten in diesem virtuellen Netz organisiert sind. In diesem *Overlay* Netzwerk lassen sich aber ebenso wie im physikalischen Netz Nachbarschafts-Beziehungen und Verbindungs-Routen durch das Netz definieren, und daraus alle üblichen primitiven Netzwerk-Operationen definieren. Dieses virtuelle Netzwerk wurde aus der Notwendigkeit heraus entwickelt, eine im Verhältnis zur Gesamtheit des Internet kleine Menge an Knoten zu organisieren.

Das Routing in einem Overlay-Netz kann sich an die physikalischen Nähe von Knoten anpassen. Dies ist aber eine Optimierung, die für ein Funktionieren des Netzes nicht unbedingt notwendig ist. Abbildung 3.1 stellt schematisch dar, wie das Routing in einem Overlay-Netz funktioniert.

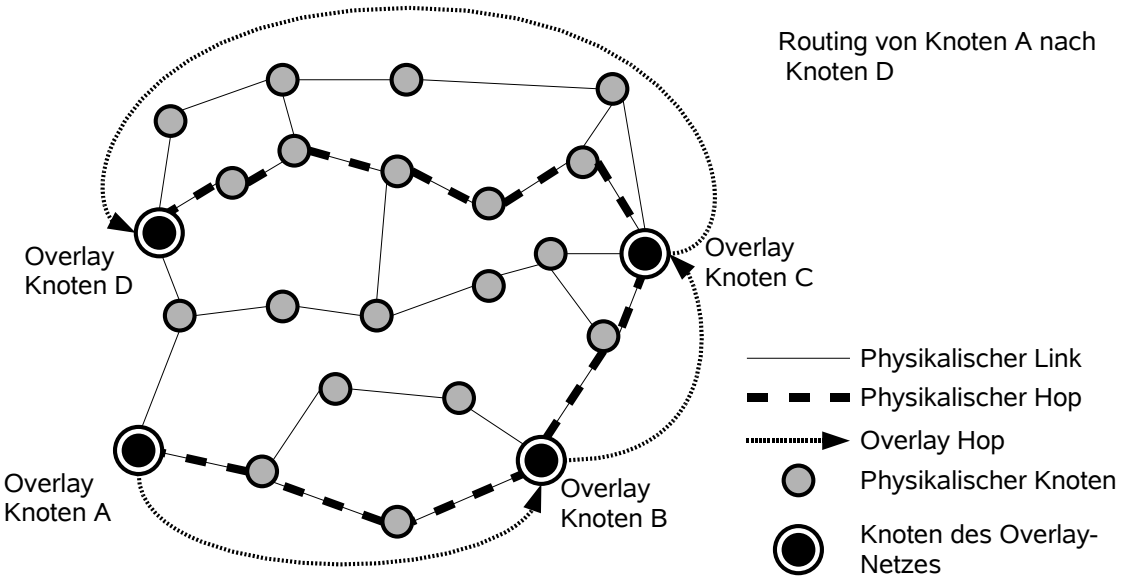

Abbildung 3.1: Overlay im Gegensatz zu physikalischem Routing (nach **?**)

Eine weitere Funktion von Overlay-Netzen ist es, verschiedene Netze zu verbinden. Dies

wird ermöglicht durch die synthetische und von den physikalischen Adressen unabhängige Addressierung der Knoten. Damit lassen sich Knoten in Netzen verbinden, die entweder nicht an das Internet angeschlossen sind, oder den Punkt der Koppelung mit dem Internet und damit die Addressierung der Knoten im Internet ändern. Beispiel hierfür ist die Anbindung von Mobilen Ad-Hoc Netzwerken an eine Infrastruktur, die das Internet für die Weitverkehrs-Koppelung nutzt.

3.2 Peer-to-Peer Systme

Peer-to-Peer (p2p) Systeme oder -Netzwerke sind eine Organisation von Knoten in einem Netzwerk für eine verteilte Anwendung, bei der es keine asymetrische Aufteilung der Aufgaben auf wenige Server und viele Clients gibt, sondern bei der alle Teilnehmer gleichberechtigt auftreten. In der Öffentlichkeit sind p2p-Systeme vor allem als Anwendung zur Softwareverteilung (Napster, BitTorrent) oder Internet-Telefonie (Skype) in Erscheinung getreten. Im Prinzip lassen sich aber nahezu alle verteilten Anwendungen nach dem p2p Prinzip organisieren.

Dabei kann man zwischen hybriden und reinen p2p-Systemen unterscheiden. In hybriden Systemen werden einzelne Aufgaben von einem oder wenigen zentralen Servern realisiert (wie zum Beispiel die Verwaltung von Metadaten zu Speicherobjekten bei Napster), und andere Aufgaben werden von den beteiligten Knoten gleichberechtigt durchgeführt (Bsp. Napster: Ablage der Daten und Übertragung zwischen den Peers).

In einem reinen p2p-Netz werden alle Aufgaben von den Peers übernommen, und es existiert kein Single-Point-of-Failure (SPoF). Damit existiert auch kein herausragender Punkt, der von einem Angreifer bevorzugt attackiert werden kann.

Darüber hinaus kann aber auch eine Verteilung der Ressourcen-Last auf alle Peers genutzt werden, um sowohl technische Flaschenhälse als auch hohe Kosten für einzelne Teilnehmer des Systems zu umgehen.

3.3 Erasure Coding

Erasure Codes dienen dazu, Redundanz zu gewährleisten, ohne den massiven Overhead einer Replikation zu erfordern. Dazu werden die Speicherobjekte in m Fragmente zerlegt und dann mittels des Erasure Codes wieder zu n Fragmenten rekombiniert, so dass $n > m$ ist.

Dabei soll $r = \frac{m}{n} < 1$ die Encoding-Rate sein. Ein Code mit der Rate r vergrößert die Speicherkosten um den Faktor $\frac{1}{m}$. Die wichtigste Eigenschaft von Erasure Codes ist die Möglichkeit, das Originalobjekt aus beliebigen m Fragmenten zu rekombinieren. Details zum mathematischen Verfahren finden sich in (**?**) und (**?**)

In (**?**) wird nun gezeigt, dass die Wahrscheinlichkeit, dass ein Speicherobjekt nicht wieder

hergestellt werden kann, bei gleichen Anforderungen an Bandbreite und Speicherkapazität, durch Erasure Coding um mehrere Größenordnungen besser ist als bei Replikation.

3.4 Kryptographie

Wotan nutzt neben dem symetrischen Verschlüsselungsverfahren AES (Advanced Encryption Standard, **?**), das asymetrische PKCS#1 Verfahren (**?**) ein. Außerdem wird massiver Gebrauch von kryptographischen Hash-Funktionen gemacht, da alle Daten mit deren Hilfe identifiziert werden.

3.4.1 Kryptographischer Hash

Ebenso wie gewöhnliche Hash-Funktionen bilden kryptographische Hash-Funktionen Eingabedaten von beliebiger Länge auf eine Bitfolge von fester Länge ab. Es gibt keine formale Definition für kryptographische Hashes, aber sie müssen gewöhnlich folgende Eigenschaften erfüllen:

- Resistenz gegen Pre-Images (I): für ein gegebenes h sollte es schwer sein ein m, zu finden, so dass gilt: $h = hash(m)$
- Resistenz gegen Pre-Images (II): für eine gegebene Eingabe m_1 sollte es schwer sein, eine andere Eingabe m_2 zu finden, so dass gilt: $m_1 \neq m_2, hash(m_1) = hash(m_2)$
- Resistenz gegen Kollisionen: es sollte schwer sein, zwei Eingaben m_1 und m_2 zu finden, für die gilt: $m_1 \neq m_2, hash(m_1) = hash(m_2)$.

Die letztere Forderung unterscheidet sich von der zweiten Pre-Image-Forderung vor allem durch die Möglichkeit (anders als bei einem fixen m_1), die Geburtstags-Eigenschaft zu nutzen, ein Verfahren zur Optimierung des Angriffes. Durch die Nutzung der Geburtstags-Eigenschaft läßt sich ein Angriff unter bestimmten Umständen um mehrere Größenordnungen beschleunigen (**?**).

Die heute meist verbreiteten kryptographischen Hashfunktionen sind MD5 und SHA-1 (**?**, **?**, **?**)

> Don't run for the door yet. But take firm steps in a doorly-direction – Bruce Schneier

Beide Algorithmen können heute nicht mehr als ausreichend sicher gelten, denn es wurde gezeigt, dass für beide mit weniger Aufwand als Brute-Force eine Kollision gefunden werden kann. (**?**, Das Papier für SHA-1 ist nur in Mandarin verfügbar). Aus diesem Grund verwendet Wotan SHA-348 als Standardvorgabe für Hashverfahren.

3.4.2 Zufallsgeneratoren

Für eine Reihe von kryptographischen Verfahren basiert die Stärke gegen kryptographische Angriffe wesentlich auf der Qualität des verwendeten Schlüsselmaterials, d.h. dass es keine innere Struktur hat und nicht aus einer vorhersagbaren Periode stammt. Daher ist es für die Erzeugung von Schlüsselmaterial wichtig, gute Generatoren von Zufallszahlen zu verwenden.

Am besten sind Generatoren die auf den Zufall in physikalischen Prozessen zurückgreifen. Klassisches Beispiel sind kleine Digitalisierer-Schaltungen, die mit Hilfe einer Z-Diode das kosmische Rauschen in einen kontinuierlichen Bit-Strom umwandeln.

Einige Betriebsysteme, wie z.B. Linux bieten die Möglichkeit, auch ohne spezielle Hardware echte Zufallszahlen zu generieren, indem sie die Länge der Intervalle zwischen dem Eintreffen von externen Ereignissen, z.B. Netzwerk-Paketen, messen und diese als Zufallszahlen nutzen. Zwar lassen sich theoretisch diese Ereignisse durch gezielte Manipulation des Netzwerkes beeinflussen, aber der Aufwand wäre sehr hoch. Darüber hinaus ist die Wahrscheinlichkeit einer Entdeckung, gerade durch die Notwendigkeit andere Einflüsse, etwas andere Paket-Quellen auszuschließen, recht hoch.

Leider ist in den meisten Java-Implementationen kein echter Zufallszahlen-Generator enthalten, und es werden auch nicht die Möglichkeiten des Betriebsystems ausreichend genutzt. Statt dess wird in der Standard-API nur ein Pseudo-Zufallszahlen-Generator verwandt, der zwar eine lange Periode liefert, aber eben doch vorhersagbare Ergebnisse. Vor allem wenn die Zahlen, die als Startwert verwandt wurden[1], dem Angreifer bekannt sind, so kann er unter bestimmten Umständen den Kreis der möglichen Schlüssel schnell stark eingrenzen.

[1] Dies ist zum Beispiel bei der Verwendung von Standardvorgaben recht wahrscheinlich

Kapitel 4

Andere Systeme

Speichersysteme sind funktionale Komponenten eines Computers seit den ersten Tagen von Relais und Röhren. Während der Ära der massiven ortsfesten Anlagen, die ganze Räume füllten, war die Anforderung an das Speichersystem simpel und offensichtlich, nämlich die Daten sicher verwahren und jedem Anwender bei Aufforderung zur Verfügung zu stellen.

Die Anforderungen in Kapitel 2 stellen eine Abkehr von dieser zentralistischen Idee dar, da sie grundsätzlich von einer Umgebung mit vielen kooperierenden Computern ausgehen, die alle zusammen arbeiten, um die ursprüngliche Anforderung zu leisten, die Daten des Anwenders sicher zu verwahren und auf Aufforderung zur Verfügung zu stellen. Es existiert eine Reihe von Ansätzen und Systemen, die einen mehr oder minder großen Teil der Anforderungen in Kapitel 2 abdecken. Das folgende Kapitel soll einen Überblick über derartige Systeme liefern und sie den Anforderungen gegenüberstellen.

4.1 Klassische Netzwerkdateisysteme

Klassische Netzwerkdateisysteme wie Network File System (NFS, **?**) oder das Common Internet File System (CIFS, **?**) leisten eine zentrale Datenhaltung und bieten die Anbindung von Speichersubsystemen an mehrere Computer, so dass von jedem Arbeitsplatz einer organisatorischen Domäne aus auf die gleichen Daten zugegriffen werden kann und der Anwender theoretisch an jedem Arbeitsplatz die gleiche Arbeitsumgebung und seine persönlichen Speicherobjekte vorfindet.

Auch transparente Zugriffe auf Web-Server (HTTP) oder FTP-Server fallen in diese Kategorie, da beide schreibenden und lesenden Zugriff auf die Daten auf dem Server bieten. Alle verbreiteten Client-Betriebsysteme wie Microsoft Windows, Linux oder Apple Macintosh X liefern einen transparenten Zugriff auf diese Server, so dass sie sich wie ein Netzwerkdateisystem verhalten.

Allerdings verfügen alle diese Protokolle über nur sehr beschränkte Discovery von Netzwerk-Ressourcen. Zwar ist im CIFS ein Finden der Server innerhalb einer Broadcast-

Domäne möglich, und die Server anderer Dienste lassen sich mit dem Service Discovery Protokoll (SDP,**?**) veröffentlichen. Beide Wege bieten aber keine Discovery über die Grenzen des lokalen Netzwerkes hinaus. Außerdem sind Berechtigungen und User-Identitäten nur auf jeweils einem Server gültig und werden nicht automatisch zwischen den organisatorischen Domänen abgeglichen.

Vor allem aber sind diese Protokolle Server-zentrisch. Die Rechtekontrolle liegt in der ausschließlichen Verantwortung des Server-Administratoren. Er kann auf jede Datei anderer Benutzer zugreifen und diese selbst beim Einsatz von Mandatory Access Controls (MAC) nach eigenem Gutdünken verändern, ohne dass der Anwender an seiner Workstation Einfluss darauf nehmen könnte. Nur mit der Verwendung von zusätzlicher Software kann der Anwender eine Ende-zu-Ende Verschlüsselung realisieren. Selbst dann geschieht dies nicht automatisch, sondern muss vom Anwender sorgfältig und manuell betrieben werden. Zu allem Übel sind Passwort-basierte Authentifikations-Methoden bei diesen Protokollen weit verbreitet. Es ist anzunehmen, dass die meisten Anwender für einen Großteil, wenn nicht alle, ihrer Authentifikationen das gleiche Passwort verwenden. Dadurch werden dem Server-Betreiber wertvolle Hilfen für einen Angriff auf andere Dienste, die der Anwender verwendet, gegeben. Eine Authentifikation des Servers gegenüber dem Anwender findet nahezu überhaupt nicht statt[1].

Die Server-Zentrierung zeigt sich auch in der Allokation von Speicherressourcen. Die Dateisysteme sind von fester Größe und können selbst bei verfügbarem Plattenplatz nicht dynamisch wachsen. Zwar kann auch hier der Server-Administrator dank Logical Volume Management und Filesystem-Resizing die Größe bei Bedarf anpassen, aber die Operationen liegen in der ausschließlichen Domäne des Administrators und stehen dem Benutzer nicht zur Verfügung.

Die Server-Zentrierung zeigt ihr größtes Problem aber in Verfügbarkeit und Lastverteilung. Der Anwender ist auf die ausschließliche Verwendung eines einzelnen Servers, im besten Falle eines Server-Clusters, angewiesen. Er kann nicht dynamisch auf andere Server ausweichen, die bessere Leistung in Bezug auf Geschwindigkeit und Netzwerk-Anbindung bieten. Auch fällt bei einer Netzwerk-Partitionierung der Zugriff auf die Daten völlig aus, da sie ausschließlich auf dem Server gespeichert werden. Damit ist das System absolut nicht für einen Einsatz in mobilen Umgebungen geeignet, wo Netzverbindungen nur für relativ kurze Zeit stabil sind und sich durch den Wechsel von Übertragungs- und Routingwegen häufig ändern können.

Das wohl wesentlichste Problem der klassischen Dateisysteme aus Sicht der Benutzerschnittstelle ist die Notwendigkeit für den Benutzer, sich an die hierarchische Verzeichnis-Struktur halten zu müssen, auch wenn dies für die Daten nicht sinnvoll ist (z.b. bei Bildern, die über einen längeren Zeitraum in festen Abständen gemacht werden). Der Name bzw. der

[1] Eine Ausnahme bilden hier Server die NFS per Kerberos authentifizieren oder HTTP über SSL verwenden, aber diese finden in lokalen Umgebungen kaum Anwendung.

Pfad der Datei legt den Speicherort fest. Der Anwender muss sich manuell um die Organisation seiner Daten in einer starren Baumstruktur kümmern und kann nicht frei wählen, nach welchen Kriterien er die Speicherobjekte bezeichnen möchte. Wird in der Arbeitsumgebung ein Protokoll wie CIFS oder auch das Andrews File System (AFS, **?**) eingesetzt, bei dem der Anwender mehrere Server dynamisch einbinden kann so muss sich der Anwender über den Namen hinaus auch den Server merken, auf dem Speicherobjekte abgelegt sind, und die Verbindung zu diesem Server unter Umständen erst manuell aufbauen.

4.1.1 Coda

Um sich dem Problem der Netzwerk-Partitionierung zu stellen, wird seit einigen Jahren am Coda System geforscht und entwickelt (Publikationen von **?**, **?** bis **?**. Coda erlaubt eine Trennung der Verbindung zwischen Server und Client und ein Weiterarbeiten des Anwenders durch den Einsatz von Caches, in denen häufig verwendete Dateien zwischengespeichert werden. Dabei sind sogar parallele Änderungen von multiplen Clients an der gleichen Datei zulässig. Um die Änderungen an Dateien nach einer erneuten Verbindung von Client(s) und Server zusammenzuführen, hat die Gruppe um M. Satyanarayanan eine Menge Arbeit in Merging-Algorithmen investiert. Sie unterscheiden mittlerweile zwischen *Transparent Merging* wie es auch aus dem Concurrent Version System (CVS, **?**) bekannt ist und *Translucent Merging*, das sich des Dateiformates bewusst ist, und bei Bedarf eine Applikation um Mithilfe beim Merging bitten kann. Für diese Mithilfe ist natürlich eine entsprechende API in der Anwendung notwendig, die vom Entwickler zur Verfügung gestellt werden muß.

Durch die enorme Menge an Dateiformaten, vor allem von proprietärem Inhalt, erscheint mir der von Coda eingeschlagene Weg als wenig erfolgversprechend, denn Coda wird nicht von einem Großen der Industrie wie Microsoft oder IBM gefördert und hat daher nach meiner Einschätzung wenig Aussicht auf die notwendige Markt-Durchdringung, um Entwickler von Dateiformaten zu einer Implementation eines Coda-Plugins zu bewegen.

4.1.2 Cluster Dateisysteme: GoogleFS, Lustre

Der Umgang mit gigantischen Datenmengen und die intelligente Verteilung der Arbeitslast sind zentrale Themen für Cluster-Filesysteme (**?**, **?**). Hinzu kommt die Problematik dass in einem System, das aus vielen tausend einzelnen Computern besteht, davon auszugehen ist, dass zu jedem Zeitpunkt ein Teil der Komponenten nicht korrekt funktioniert.

Diese Themen werden durch eine Trennung von Daten und Metadaten in Angriff genommen. Dabei geschieht die Übertragung der Nutzdaten direkt zwischen dedizierten Speicherknoten und dem generierenden Knoten. Ein Metadaten-Server koordiniert nur diee Übertragungen und hat ein um drei bis vier Größenordnungen geringeres Datenvolumen als das Gesamtsystem zu handhaben. Dies macht auch eine Replikation dieser zentralen Komponente möglich, so dass eine Redundanz im Falle eines Ausfalls zur Verfügung steht. Die

Konsistenz der Nutzdaten wird durch Speicherung von Paritäts-Informationen erreicht, dem RAID-5 nicht unähnlich.

Ein Nachteil, der auch in das Blickfeld der Entwickler von Lustre gerutscht ist, ist der Mangel an automatischer Konfiguration. Dies ist für den Anwendungszweck nicht zu gravierend, da fast alle beteiligten Knoten gleichartig sind, so dass eine einzelne Konfiguration auf alle Knoten des Systems kopiert werden kann. Dennoch soll dieser Mangel abgestellt werden, um den allgemeinen Konfigurationsaufwand eines High-Performance-Clusters zu reduzieren. Aufgrund ständig fallender Hardware-Preise hat sich die Konfiguration als Kostentreiber einer HPC-Installation herausbildet.

Die meist geschlossene Umgebung und die Begrenzung des Zugriffs auf den Cluster durch eine sog. Headnode machen einen Schutz von Dateien gegen unberechtigte Zugriffe innerhalb des Systems überflüssig. Daher verfügen die Cluster-Dateisysteme über keine E2EE (siehe Abschnitt 2.2.3).

4.1.3 Kooperative Front-Ends: SiRius

In SiRius (**?**) wird die Möglichkeit untersucht, mit einem sicheren Front-End auch Back-Ends, denen nicht vertraut werden kann, zu unterstützen. Dies wird in erster Linie durch eine Ende-zu-Ende Verschlüsselung erreicht, liefert aber auch die Möglichkeit, unterschiedliche Back-Ends innerhalb einer Umgebung zu unterstützen. Dabei werden nicht nur klassische Netzwerkdateisysteme wie NFS oder SMB untersucht, sondern auch web-basierte Speichersysteme wie Yahoo!.

Ein zentraler Punkt der oben genannten Anforderungen aber, nämlich die automatische Entdeckung von Speicherorten, wird bei SiRius nicht betrachtet und auch nicht unterstützt. Auch betrachtet der Ansatz nicht die Frage nach Replikation durch das Backend, sondern erwartet einfach, dass ein angemessenes Backup durch die Betreiber der Back-Ends unterstützt wird.

4.2 Relationale Datenbanken

Gegenüber den Netzwerk-Dateisystemen stellen Relationale Datenbank Management Systeme (RDBMS) [2] das andere Extrem bei der der Organisation von Daten dar. Hier kann der Anwender seine Daten nach Belieben organisieren und der Software auch Abhängigkeiten zwischen Daten bekannt machen, so dass diese über die Einhaltung wachen kann. Jedoch muss dies vor der Verwendung geschehen und ist Laien ohne SQL Kenntnisse nur selten möglich.

[2] Darunter sind für dieses Dokument auch objekt-relationale und sogar objektorientierte Datenbank-Management Systeme zusammengefasst, da sie alle eine (wenigsten weitgehende) Apriori Definition der Struktur der Datenbank erfordern

Die hohe Strukturierung der Daten erlaubt ein einfaches Merging von Änderungen nach einer Netz-Partitionierung oder einem Offline-Betrieb. Im praktischen Einsatz bieten kommerzielle Produkte, nach einer geringfügigen Anpassung des Datenmodells, eine transparente Zusammenführung von Änderungen. Da die gesamte Struktur der Daten bekannt ist, gibt es keine Probleme mit proprietären Dateiformaten[3].

Aber RDBMS teilen mit den Netzwerk-Dateisystemen die Server-Zentrierung. Zwar gibt es allerlei Technologien und Produkte, um die Server zu Clustern zusammenzufassen oder zu partitionieren, so dass Server-Ressourcen in der Ausfallsicherheit erhöht werden können und automatisch Speicherorte physikalisch in die Nähe des Anwenders bewegt werden können. Aber dennoch fallen alle Speicher-Ressourcen in eine einzige administrative Domäne, in der es einen Administrator mit vollständigen Rechten gibt. Dadurch wird auch bei einem RDBMS die Forderung nach Ende-zu-Ende Verschlüsselung verletzt. Client-Anwendungen, die eine Verschlüsselung unterstützen, würden durch die fehlende Offenlegung der Datenstruktur den Vorteil eines RDBMS aufheben.

4.3 Peer-to-Peer Netzwerke

Peer-to-Peer (P2P) Netzwerke versprechen mit einer wesentlichen Eigenschaft anderer Speichermethoden zu brechen: der Bindung des Namens eines Speicherobjektes an seinen Speicherort. Peer-to-Peer Netzwerke automatisieren zu einem mehr oder minder weitem Grad die Suche nach Speicherorten für ein bestimmtes Speicherobjekt. Diese Speicherorte können mehrere Server sein und sind in den meisten Fällen auch über weitere Server repliziert.

Durch die Gestaltung ihrer Anwender-Clients, Detail-Lösungen der Protokolle, durch Vorlieben der Anwender oder organisatorische Vorgaben degradieren die meisten Peer-2-Peer Anwendungen allerdings zu *Distributions-Systemen*, die lediglich von einer einzigen Quelle an viele Clients (1-to-n) verteilen, denn wirkliche Peer-2-Peer *Austausch-Systeme* (n-to-m) zu sein. Nach meiner Ansicht ist dies vor allem in der Lernkurve begründet, die für einen Anbieter von Daten steiler ist, als für eine Verbraucher von Daten. Damit verletzen die meisten dieser Anwendungen die Forderung nach möglichst einfacher Benutzung durch den Anwender.

4.3.1 Napster

Als Vertreter der ersten Generation von P2P Netzwerken verwaltete Napster (**?**) die Speicherorte auf einem zentralen Server (bzw. einem Cluster von Servern). Diesem Server meldeten die Peers die Speicherobjekte, die sie verwalteten und auf Anfrage gab der Server diese Information an andere Peers weiter.

[3] Voraussetzung hierfür ist, dass die Daten einzelner Felder keine innere Struktur haben

Diese Modell hat zwei wesentliche Nachteile. Zum einen basiert es auf einem zentralen Server, der sich als Ziel für Angriffe eignet, da das gesamte System ohne diesen Server zusammenbricht. Wertet man Napster als System-Modell, so meint ein Angriff gewöhnlich eine technische Maßnahme, die zu einem Verlust der Leistung des Systems führt. Im Falle von Napster kam der Angriff aber auch als juristische Klage gegen den Betreiber des Servers, da beim Austausch von Musik- und Filmdateien über das System große Mengen von Verstößen gegen Copyright-, Urheberrechts- und Verwertungsgesetze geschahen. Der zentrale Server konnte so durch eine gerichtliche Anordnung abgeschaltet werden und damit das gesamte System lahmgelegt werden.

Der andere Nachteil ist das Fehlen einer Schreib-Operation. Zwar können Speicherobjekte in das System eingestellt werden, deren Existenz dann vom Peer an den Server gemeldet welden. Aber die Daten werden erst auf andere Computer übertragen, wenn andere Peers eine Lese-Operation auf genau dieses Speicherobjekt beginnen. Die Veröffentlichung der Information, dass diese Datei existiert, muss mit einem Out-of-Band Mechanismus geschehen. Damit gibt für den Erzeuger des Speicherobjektes keinen Weg sicherzustellen, dass andere Peers eine Kopie der Daten angelegt haben.

4.3.2 BitTorrent

BitTorrent (**?**)ist der am weitesten verbreitete Vertreter der P2P-Netze der zweiten Generation. Zu dieser Gruppe von P2P-Netzen gehören auch eDonkey (**?**), Gnutella (**?**) und weitere. Diese Netze zeichnet vor allem das Fehlen eines zentralen Servers aus. Und während bei eDonkey und Gnutella einige Knoten besondere Eigenschaften haben[4], grundsätzlich aber gleich funktionieren, unterscheidet BitTorrent zwischen Peers und Trackers.

Trackers verwalten die Peers, welche Teile eines Speicherobjektes zur Verfügung stellen. Die Peers, die entweder Teile des Speicherobjektes haben oder erhalten wollen, melden sich beim Tracker und erfahren von diesem, welche Peers über welche Elemente des Speicherobjektes verfügen. Damit stellen die Tracker natürlich einen Single-Point-of-Failure (SPoF) dar, allerdings nur für die Speicherobjekte die von diesem Tracker verwaltet werden. Da jeder Anwender seine eigenen Tracker starten kann, ist dies für das Gesamtsystem nicht besonders bedrohlich.

Doch auch diesen SPoF wollen die Entwickler entfernen und haben daher mit der Entwicklung des sog. Trackerless-BitTorrent begonnen. Dabei handelt es sich um eine Verwaltung der Tracker-Informationen mit einer DHT. Damit wäre BitTorrent völlig unabhängig von zentralen oder ausgezeichneten Knoten.

Aber auch BitTorrent in seiner augenblicklichen Gestalt erfüllt noch nicht die Anforderungen, wie sie in Kapitel 2 definiert sind. Zum einen muss der Anwender, selbst bei Verwendung des Trackerless-BitTorrent, noch einige andere BitTorrent Knoten kennen, um Zugang zum

[4] "Alle sind gleich, aber einige sind gleicher" – George Orwell, Animal Farm

Overlay-Netz zu erhalten. Und die Gestaltung der notwendigen Software als Anwendungsprogramm statt als Infrastruktur-Komponente macht es unwahrscheinlich, dass die Anzahl der BitTorrent-Clients weit über die Anzahl der Anwender hinaus wächst. Ein flächige Durchdringung der Internet-Knoten oder wenigstens der Broadcast-Domains mit jeweils mindestens einem BitTorrent-Client ist daher eher unwahrscheinlich.

BitTorrent hat für die Lastverteilung bei der Distribution von Speicherobjekten bewährt. Durch die Anzahl von organisatorisch unabhängigen Peers bietet es auch eine hohe Stabilität gegen Datenverlust. Allerdings steuert auch bei BitTorrent, wie bei allen anderen P2P Anwendungen, vor allem das Interesse der Anwender an einem Speicherobjekt, die Stabilität, indem sie möglichst viele lokale Kopien anlegen.

So liefert BitTorrent ebensowenig wie andere P2P Programme die Möglichkeit, schreibend Speicherobjekte in das System einzustellen, noch E2EE oder noch wirkliche Discovery.

4.3.3 Mute

Erwähnenswert ist hier noch ist Mute (**?**), bei dem das Routing zwischen den Peers laut Beschreibung nicht verfolgt werden kann und damit die Anonymität der Teilnehmer gewährleistet wird. Das Konzept des Routings basiert auf der Beobachtung von Ameisen, die bei einer Erkundung ihrer Umgebung wenig zielgerichtet alles untersuchen. Finden sie Nahrung, so kehren sie mit einer Probe zum Bau zurück. Dabei hinterlassen sie eine Duftspur, die andere Ameisen nutzen, um auf direktem Weg zur Nahrungsquelle zu kommen.

In Mute dienen alle beteiligten Computer als Router in einem Overlay-Netzwerk. Durch bestimmte Maßnahmen kann von außen, ohne direkten Zugriff auf den Computer, nicht festgestellt werden, ob ein Datenpaket weitergeleitet wird oder für den jeweiligen Computer bestimmt war. Damit kann der Besitzer eines Computers immer argumentieren, lediglich Daten weitergeleitet zu haben ohne von deren Inhalt Kenntnis genommen zu haben.

Leider fehlt Mute eine Technik, um vorhandene Speicherobjekte oder Hinweise auf Inhalte zu publizieren. Im Moment werden Foren und IRC für diesen Zwecke benutzt, wodurch der Vorteil der Anonymität wieder eingebüßt wird.

4.3.4 Chord

Chord ist eine DHT mit dem Design-Fokus eines verteilten Lookups, das sich aus Anwendersicht ähnlich zum Domain Name Service (DNS) verhält: es speichert das Mapping eines Schlüssels zu einem Wert, und dabei ist die Datenbasis auf verschiedene Knoten verteilt. Durch Replikation und eine ringförmige *successor*-Beziehung ist ein Auffinden des Knotens, der die gesuchte Information enthält, fast immer gegeben, selbst wenn Knoten arbiträr die DHT verlassen und wieder betreten.

Chord selbst liefert keine Verwaltung von Daten, dient aber als Basis für eine Reihe von Diensten von höherer Komplexität und wird auch in Wotan als Lookup-Dienst eingesetzt.

4.3.4.1 DHash

DHash (**?**) ist ein verteiltes Daten-Verwaltungssystem auf der Basis von Chord, das Blöcke von Daten mit Hilfe von Erasure Coding gegen Datenverlust schützt und dann auf die Knoten eines Chord Overlay-Netzes verteilt. DHash beinhaltet außerdem ein Protokoll, um beim Verlust von Knoten die Informationen, die auf diesen Knoten abgelegt waren, aus den noch vorhandenen Fragmenten zu rekonstruieren und auf noch vorhandene Knoten zu verteilen. Außerdem werden beim Beitritt von neuen Knoten in das System Fragmente weitergeleitet, um eine möglichst gleichmäßige Auslastung der Speicherressourcen und Verteilung der Netzwerklast zu erreichen.

DHash ist als Basis für komplexere Systeme gedacht und konzentriert sich mit seiner einfachen API ausschließlich auf die Verwaltung von Daten-Blöcken. Alle anderen Aufgaben wie E2EE oder Discovery sind ausgeklammert.

Auch wenn der Fokus der Entwickler von DHash auf einem verteilten Archivsystem lag, so erfüllt DHash schon einen Großteil der Anforderungen, die an Wotan gestellt sind, oder kann mit relativ wenig Aufwand um diese Funktionen erweitert werden. Allerdings macht DHash die Annahme, dass alle Knoten nahezu kontinuierlich erreichbar sind. Es wird angenommen, dass zu jedem Zeitpunkt nur ein sehr kleiner Teil der Knoten durch Netzwerk-Partionierung vom System getrennt wird, so dass die Wartungs-Protokolle die Datenblöcke rekonstruieren können.

4.3.4.2 Cooperative File System (CFS)

Das CFS (**?** und **?**) ist ein Beispiel für ein vollständiges UNIX-Dateisystem, das von der Arbeitsgruppe stammt, die auch Chord und DHash entwickelt haben. Es ist ein Read-Only Dateisystem, in dem der Mechanismus des Publizierens außerhalb von CFS liegt. Beim Publizieren, dem Einstellen von Dateien in die Menge der verfügbaren Dateien, wird die Datei in ein bestimmtes Verzeichnis gelegt und ist von nun an für Suchen sichtbar

Durch den DHash-Mechanismus, der CFS zu Grunde liegt, werden die Daten auf die verschiedenen Computer verteilt und bei einem Zugriff vom Client geladen. Damit erfüllt CFS die gleichen Anforderungen und Annahmen wie DHash bei der Stabilität gegen den Ausfall von Knoten und Netzwerkpartitionierungen.

CFS ist für Wotan vor allem als Vorlage für ein vollständiges System interessant. Es kann als Messlatte sowohl für die Bequemlichkeit der Anwendung, als auch für Antwortzeiten und Bandbreiten dienen.

Ebenso wie DHash liefert auch CFS weder Discovery von lokalen Knoten noch E2EE. Auch ist das zur Verfügungstellen von Speicherkapazität und Bandbreite ein freiwilliger Vorgang, und es gibt kein Konzept, wie diese Leistungen abgerechnet werden können.

Das Hauptziel von CFS war es, neben der Lastverteilung, wie DHash sie liefert, auch die Verwendung durch den Anwender möglichst einfach zu gestalten. Daher ist es aus Sicht

des Anwenders als klassisches Dateisystem implementiert, das sich unter Linux wie jedes andere Read-Only Dateisystem (CDs und DVDs zum Beispiel) einsetzen lässt. Im Fazit zu CFS wird allerdings auch die Aussage gemacht, das CFS als Beispielanwendung gedacht war, um einmal zu überprüfen, wie so eine Datenverwaltung auf Basis eines P2P Systems aussehen könnte, und dass die Autoren hoffen, dass andere die Mängel von CFS aufgreifen und abstellen könnten. Unter anderem die Stabilität des Systems gegen bösartige Knoten, welche das System beeinträchtigen wollen, lasse noch zu wünschen übrig und erfordere noch weitere Forschungen.

4.3.5 Pastry

Pastry (**?**) ist ebenso wie Chord eine Implementation für eine DHT. Pastry unterscheidet sich von Chord vor allem durch eine hierarchische Routing-Tabelle, die aus dem Wissen über die Entfernung von einem Knoten zu einem Zielknoten eine Routing-Optimierung durchführen kann. Darüber hinaus verwaltet ein Pastry-Knoten auch eine Liste von weiteren Knoten als Blätter wie in einem B-Baum. All dies führt zu einer Optimierung des Aufwands für das Finden eines Zielknotens. Allerdings erzeugt dies natürlich einen gewissen Overhead bei der Pflege der Routing-Informationen.

In dem Papier, in dem Pastry vorgestellt wird, macht der Autor explizit die grundsätzliche Annahme, dass ein gewisser Satz an entfernten Knoten jedem Knoten apriori bekannt ist. Die meisten anderen Papiere zu DHTs machen diese Annahme stillschweigend. Eine lokale Discovery oder die Verteilung von Informationen zu geeigneten Startknoten ist nicht Bestandteil von Pastry. Ebenso wie Chord ist Pastry nur als Grundlage für komplexere Dienste gedacht.

Zielumgebung von Pastry ist das normale Internet. Daher kann von einem stabilen und verlässlichen Routing in der Netzwerk-Ebene ausgegangen werden. Wotan ist vor allem auch für Mobile AdHoc Netzwerke (MANet) gedacht und muss daher eine DHT mit entsprechenden Eigenschaften haben. Interessant ist Pastry aber vor allem wegen der Forschungen um MADPastry (**?**), die ein mobiles Szenario als Umgebung für Pastry überprüfen und daraufhin zu optimieren suchen.

4.3.5.1 PAST

Auf Pastry aufbauend wurde in (**?**) PAST vorgestellt. Es ist ebenso wie DHash ein verteiltes Archivsystem und verfolgt ähnliche Ziele. Ein wichtiger Unterschied zu Wotan ist die explizite Annahme, dass Speicherobjekte sich als Inhalte eines Archives nie ändern.

Auch in Wotan können Inhalte auf der Ebene eines Speicherobjektes nicht geändert werden, sondern eine Änderung führt zu einem neuen Objekt. Aber auf der darunterliegenden Ebene der Dateiblöcke können unveränderte Dateiblöcke, die vor der Änderung und damit dem Anlegen des neuen Objektes bereits im System vorhanden waren, weiter verwendet

werden, ohne erneut verteilt werden zu müssen. In der Beschreibung des GoogleFS (**?**) werden die Optimierungspotentiale einer besonderen Behandlung von `append()` Operationen gesondert behandelt.

In PAST wird der Identifikator für ein Speicherobjekt aus Namen, Owner-Credential und einem Salt generiert und nicht durch Inhalt der Datei bestimmt. Mögliche Kollisionen werden nachträglich detektiert und erfordern die Kenntnis des Systems aller bereits vergebenen Namen. Die erforderliche Synchronisation der Namensliste macht damit eine unabhängige und nebenläufige Verteilung des Systems unmöglich.

4.3.6 OceanStore

OceanStore (**?**) ist genau wie PAST (Abschnitt 4.3.5.1) ein Speichersystem, das auf einer DHT-Infrastruktur aufsetzt. Die Überlegungen, die zur Entwicklung von OceanStore geführt haben (ubiquitäre Geräte, Skalierbarkeit, Anforderungen an Robustheit des Gesamtsystems), sind mit denen von Wotan sehr eng verwendet.

Aber OceanStore nimmt ebenso wie PAST eine verfügbare Infrastruktur mit ausreichender Bandbreite an (in der aggregierten Bandbreite quasi unbegrenzt). OceanStore macht keine Optimierungen für MANets und begrenzte Bandbreiten. Auch ist es nicht auf kooperative Speicherung im soziologischen Sinne ausgerichtet. Es trennt zwischen Anwendern und Dienstanbietern, die sich die angebotenen Dienste in realem Geld bezahlen lassen. In OceanStore stellen Anwender keine Ressourcen für das Allgemeinwohl zur Verfügung. In Wotan tun dies die Anwender, nicht aus altruistischen Gründen, sondern um bei Bedarf selbst auf Ressourcen anderer Anwender zurückgreifen zu können.

Noch grundsätzlicher ist der Unterschied, das Datenobjekte in OceanStore durch das Versenden von Updates geändert werden können. Auch abstrahiert OceanStore nicht vollständig von Anwender-definierten Metadaten. Stattdessen nutzt es einen vom Erzeuger des Datenobjektes vergebenen Namen zusammen mit dem öffentlichen Schlüssel das Anwenders zur Generierung eines eindeutigen Identifikators für Datenobjekte.

Kapitel 5

Architektur von Wotan

5.1 Lösungsansatz: Kooperative Speicherung

Während der Analyse von Fehlerfällen in Systemen zur Datenspeicherung haben mehrere Autoren (z.B. **?**,**?**) die Trennung der Sicht auf die Datenspeicherung aus Sicht des Anwenders und aus Sicht des Betreibers des Datenspeichers untersucht. Sind Anwender und Betreiber/Besitzer des Speichers unterschiedlich, so verliert in der Regel der Anwender einen Teil der Kontrolle über die Art und Weise, in der seine Daten abgelegt sind. Daraus ergibt sich folgende Definition für Kooperative Speicherung:

> Kooperative Speicherung ist jede Form der Datenspeicherung, bei welcher der Anwender keine oder nur unvollständige Kontrolle über das Speichermedium ausübt.

Dabei beschreibt "kooperativ" nicht die gesellschaftliche oder rechtliche Relation von Dateneigentümer und Betreiber des Speichers. Diese können durch externe Regelungen oder Verträge in einem Zwangsverhältnis stehen. Einzig der Mangel an vollständiger Kontrolle über die Art und Weise der Speicherung durch den Anwender dient als Abgrenzung, ob eine kooperative Speicherung vorliegt oder nicht.

Kooperative Speicherung tritt in einer Reihe von Anwendungsfällen aus unterschiedlichen Bereichen auf:

Anwendungen mit Mandatory Access Controls (MAC) In Bereichen, in denen MACs (wie etwa nach dem Bell/LaPadula Modell (**?**) z.B im militärischen Umfeld) eingesetzt werden, muss der Administrator eines klassischen Systems mindestens so hohe Freigaben haben wie der Anwender mit der höchsten Freigabe. Da dies aus praktischen Gründen selten möglich ist (der General darf alles lesen, aber nicht der Gefreite, der den Computer wartet), kann die Anforderung als Form der kooperativen Speicherung betrachtet werden.

Aus diesem Anwendungsbereich ergibt sich noch deutlicher als aus anderen Bereichen die Anforderung nach E2EE (siehe Abschnitt 2.2.3). Nur mit Hilfe von E2EE lassen sich MACs auch in Umgebungen mit kooperativer Speicherung durchsetzen.

Outsourcing Dies ist wohl der zur Zeit kommerziell relevanteste Bereich zum Thema kooperative Speicherung. Gleichgültig wie streng der Vertrag zwischen den Partnern ist, so ist doch die obige Definition erfüllt. Ein System, das Sicherheit und Stabilität liefert und die Konfigurationsaufgaben minimiert oder auf den Anbieter des Speicherplatzes abwälzt, ist kommerziell relevant. Für den Anbieter des Speicherdienstes ist es natürlich ebenso interessant, minimale Kostenaufwände durch die Konfiguration seiner Systeme zu haben und durch eine dokumentierbare Mandantenfähigkeit die Daten seiner unterschiedlichen Kunden voneinander zu trennen.

High Performance Computing (HPC) Während in diesem Bereich sowohl die Daten als auch die Speichergeräte unter der Kontrolle einer Organisation stehen, ergibt sich aus der enormen Anzahl der beteiligten Komponenten ein ähnliches Problem wie in Ubiquitous Computing Umgebungen. Das überwiegende Design-Paradigma in diesem Bereich der Informatik ist zur Zeit der Cluster: Ein Zusammenschalten von vielen gleichartigen Computern, meistens aus sog. COTS[1] Komponenten. Hier ist es nicht möglich, in akzeptabler Zeit die manuelle Konfiguration aller Einzelteile zu leisten.

Grid-Computing Das grundlegende Problem der Discovery und Konfiguration von Ressourcen verschlimmert sich natürlich, wenn die einzelnen Elemente in unterschiedlichen Organisationen und administrativen Domänen beheimatet sind. Das Grid-Computing verspricht aber gerade den Anwendern über diese Grenzen hinweg Rechen- und Speicherleistung zu liefern. Die heute verfügbaren Ansätze (**?**) sind entweder nicht in der Lage, dies vollständig transparent zu leisten oder wälzen die Arbeit auf den Administrator ab (**?**, Kap. 9).

Ad-Hoc Communities Zusammenschluß von Computern, die durch ihre Anwender in die Reichweiter der installierten Netzwerk-Unterstützung (Bluetooth oder WLAN) getragen werden. Es entstehen je nach Gruppenbildung der Anwender dynamisch neue Netzwerk-Strukturen der Geräte. Besonder deutlich zeigt sich dieses Einsatz-Szenario bei Geräten ganz ohne Benutzerschnittstelle wie dem “Personal Data Stick”[2](PDS). Das Gerät speichert alle Daten, die von kompatiblen Diensten publiziert werden und in deren Reichweite es vom Besitzer getragen wird.

Die Speicherobjekte werden in diesem Anwendungsfall zwar bewußt vom Erzeuger publiziert, der Benutzer muß aber nicht aktiv eingreifen, um die Daten auf seinen Spei-

[1] Component of the shelf: Handelsübliche Geräte

[2] Das Gerät ist im Prinzip ein Bluetooth-Dongle mit Batterie, vor ein paar Jahren von Hewlett-Packard vorgestellt.

cher zu laden. Weder der Erzeuger der Daten noch der Betreiber des Speichermediums können direkt kontrollieren, wann welche Daten übertragen werden. Dies ist aus dem Gesichtspunkt der Ressourcenverwaltung ein interessanter Anwendungsfall, da die Frage geklärt werden muß, wer wieviele Informationen auf einen dieser PDS schreiben darf.

5.2 Design Philosophie

Wotan arbeitet grundsätzlich nach dem “Best Effort” Prinzip und versucht in der überwiegenden Mehrzahl der Fälle für den Benutzer das Richtige zu tun, und bei Fehlerfällen den Benutzer geeignet zu informieren. Dieser Grundansatz ist auch bei anderen Protokollen wie zum Beispiel dem IP-Protokoll zu finden.

Wotan versucht auch in der Mehrzahl der Fälle auf eine Anfrage des Benutzers vorbereitet zu sein (z.B. indem die Chunks eines Files alle geladen werden, wenn das File das erste Mal geöffnet wird.) und Kontakt zu anderen Knoten zu nutzen, um zusätzliche Informationen über Speicherobjekte zu erlangen, wenn sich gerade die Gelegenheit bietet.

Ist eine Information wie ein Datei-Chunk oder ein Schlüssel beim Aufruf durch den Benutzer nicht verfügbar, so wird ein erneuter Versuch gestartet, diese Informationen zu erhalten. Ist dies in vernünftiger Zeit[3] nicht möglich, so antwortet Wotan mit einem Fehler auf die Anfrage.

Die zugrunde liegene Annahme ist die Überlegung, dass ein massiv verteiltes System nicht deterministisch funktionieren kann, selbst wenn alle möglichen Fehlerzustände der Knoten ermittelt werden könnten. Der Zeitpunkt des Auftretens dieser Fehlerzustände kann jedoch nicht vorhergesagt werden. Als Reaktion darauf kann nun entweder zusätzlicher Aufwand betrieben werden, um das System für den Anwender deterministisch erscheinen zu lassen[4], oder nach dem “Best Effort” Ansatz nach einem Ausschöpfen der Möglichkeiten den Anwender über den Zustand informieren.

5.3 Überblick

Wotan dient der Speicherung von beliebigen Daten in Speicherobjekten von beliebiger Größe. Der Ort der Speicherung und der Weg des Transportes von der Anwendung, die ein Speicherobjekt erzeugt zur Anwendung, welche die Daten des Objektes aufruft, sind für den Benutzer vollkommen transparent. Einen Überblick über die Arten von Knoten im Wotan Netzwerk und der Datenpfade zwischen den Knoten gibt die Abbildung 5.1.

[3] Ein subjektiver Begriff, der evtl. durch den Benutzer gesetzt werden kann

[4] Die Reaktion des TCP Protokolls auf Paketverlust ist ein bekanntes Beispiel für diesen Ansatz

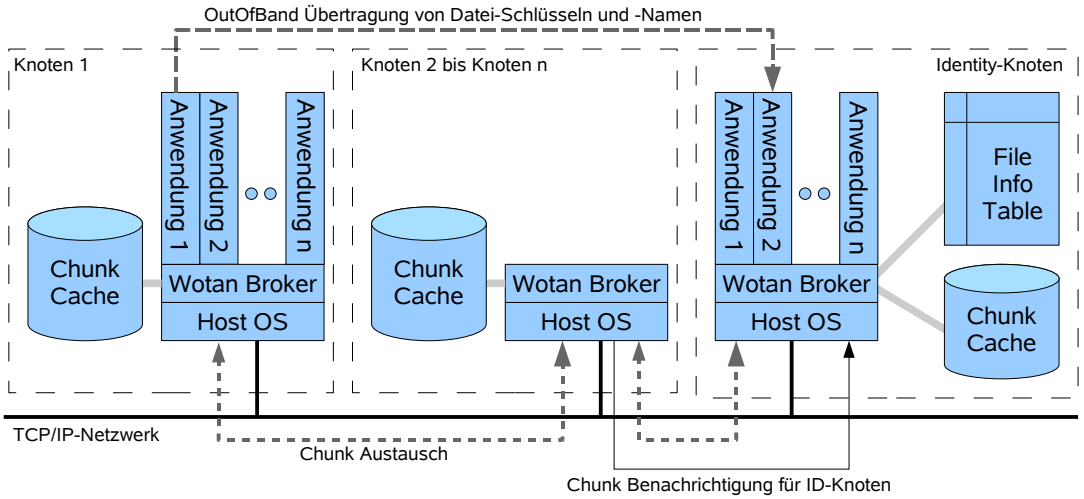

Abbildung 5.1: Arten von Knoten in Wotan

Speicherobjekte werden auf einem Computer erzeugt und von diesem nach eigenem Ermessen an andere Computer, die an Wotan angeschlossen sind, weitergeleitet. Jedesmal, wenn das Objekt einen neuen Computer erreicht hat, benachrichtigt dieser die Computer des Anwenders durch eine Botschaft über das DHT-Overlay Netzwerk, dass nun eine Kopie des Speicherobjektes auf einem weiteren Computer verfügbar ist. Dabei ist der Benutzer nicht auf einen bestimmten Computer beschränkt, sondern alle Computer, die eine Kopie der Verwaltungsinformationen für die Speicherobjekte besitzen, erhalten auch eine Kopie der Aktualisierungsnachricht.

Dabei ist ein Computer unter Umständen nicht in der Lage oder willens dem Benutzer eine Aktualisierungsnachricht zu senden. Dies wird auch nicht durch das Protokoll erzwungen. Das Einsatz-Szenario von Wotan geht davon aus, dass die Knoten sich nicht in der gleichen organisatorischen Domäne befinden, und das nicht nur eine kleine Anzahl von solchen Domänen beteiligt ist. Daher ist eine Behandlung von Leistungserbringung und -abrechnung zwischen den Knoten ein notwendiger Teil des Wotan Gesamtkonzeptes. Sozialer Druck kann hier nicht ausreichen.

Die Computer des Anwenders pflegen Listen mit Speicherorten für die Daten des Anwenders. Möchte der Anwender auf die Daten zugreifen, kontaktiert der von ihm verwendete Computer diese Speicherorte und lädt die Speicherobjekte. Die Namen der Speicherobjekte, sowie die geheimen Schlüssel, mit denen alle Speicherobjekte verschlüsselt sind, müssen mit einem Verfahren auf den oder die Computer des Anwenders gebracht werden, das außerhalb von Wotan liegt. Die Namen der Speicherobjekte sind nicht für einen menschlichen Gebrauch bestimmt sind, da sie mit Hilfe von kryptographischen Hashfunktionen aus dem Inhalt der Speicherobjekte generiert werden. Im Folgenden wird davon ausgegangen, dass

diese Informationen zur Verfügung stehen, wenn der Anwender auf ein Speicherobjekt zugreifen möchte.

Wotan unterscheidet sich von anderen verteilten Speichersystemen durch zwei wesentliche Eigenschaften: Die Rückwärts-Notifikation und die Weiterleitung der Daten durch Austausch zwischen jeweils zwei Knoten, in einer Art Store-and-Forward Verfahren. Während Abschnitt 5.8 die Weiterleitung näher beschreibt, wird die Rückwärtsnotifikation im nächsten Abschnitt dargestelt.

Außerdem stellt Wotan eine Abkehr von der Desktop-Metapher mit seinen (virtuellen, digitalisierten) Aktenschränken und Hängeordnern. Stattdessen trennt es konsequent die Verwaltung der Metadaten (die in Wotan vollständig ausgeklammert werden) von der Verwaltung einer großen Anzahl von Speicherobjekten auf einer dynamischen Menge von Geräten mit Rechen- und Speicherleistung.

5.3.1 Rückwärts-Notifikation

Andere verteilte Speichersysteme (z.b CFS, Abschnitt 4.3.4.2 oder PAST, Abschnitt 4.3.5.1 , die auf DHTs beruhen, verteilen gezielt die Daten auf andere Knoten und verwalten dann die Informationen über die Speicherorte mit Hilfe des DHT. Dafür ist aber apriori das Wissen notwendig, welche Knoten sich an dem Speichersystem beteiligen. Zumindest müssen sich die beteiligten Knoten zum Zeitpunkt der Ablage des Speicherobjektes in das DHT eingetragen haben. Diese Annahme ist für ein relativ stabiles Netzwerk wie das Internet auch korrekt.

Wotan zielt aber auf eine Umgebung ab, in der Netzpartitionierung zur Normalität gehört. Daher setzt Wotan auch auf eine Store-and-Forward Methodik bei der Weiterleitung von Speicherobjekten, und schließt ausdrücklich die Annahme aus, dass zu jedem Zeitpunkt jeder Knoten alle oder auch nur einen Großteil der beteiligten, Knoten erreichen kann.

Um nun dennoch die Speicherorte der Speicherobjekte zur Verfügung zu haben, wenn ein Anwender auf seine Daten zugreifen möchte, nutzt Wotan ein Konzept namens *Rückwärts-Notifikation*. Wenn während der Verteilung der Speicherobjekte diese auf einem beliebigen Knoten A ankommen, kann dieser Knoten den Eigentümer bzw. seine Computer benachrichtigen, dass die Daten auf Knoten A zur Verfügung stehen. Diese Nachricht bezieht sich nur auf den augenblicklichen Zustand. Wenn die Daten dann weitergeleitet werden, wird der nächste Knoten B, der die Speicherobjekte entgegennimmt, erneut den Anwender über den neuen Speicherort seiner Daten benachrichtigen.

Die Anahme dabei ist, dass sich von den Computern eines Anwenders mindestens einer (sein persönlicher PC, sein Home-Server oder ein beauftragter externer Dienst) zu jeder Zeit in der Lage befindet, auf die Nachrichten zu reagieren und eine Liste der Speicherorte zu pflegen. Dabei dienen DHTs dazu, die Liste der Computer zu verwalten, die an den Nachrichten der Rückwärts-Notifikation für ein bestimmtes Speicherobjekt interessiert sind.

5.4 Bausteine

5.4.1 File und Chunk

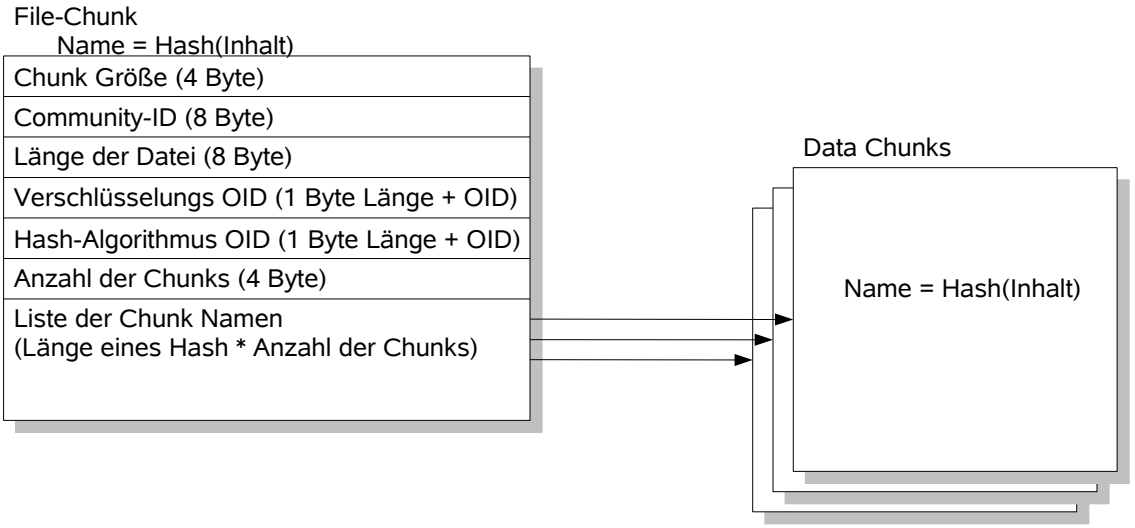

Abbildung 5.2: Aufbau von Dateien in Wotan

Daten in Wotan werden in Blöcken, den sogenannten *Chunks* verwaltet. Die Chunks können zwar im Prinzip von beliebiger Größe sein, aber innerhalb eines Kontextes ist die Größe aller Chunks festgelegt.

Der identifizierende Name dieser Chunks ermittelt sich aus ihrem Inhalt. Dazu wird über den Inhalt ein kryptographischer Hash (Abschnitt 3.4.1) berechnet. Damit wird der Chunk eindeutig identifizierbar, ohne dass vom Namen auf den Inhalt geschlossen werden kann. Eine Änderung des Inhaltes hätte darüber hinaus eine Änderung des Namen zur Folge.

Das für den Anwender sichtbare Speicherobjekt ist das *File*. Das File dient der Organisation von Chunks und beinhaltet, wie eine I-Node im Unix Dateisystem, alle Informationen zu diesem Speicherobjekt (mit Ausnahme des Schlüssels, mit dem die Daten in den Chunks verschlüsselt sind). Abgelegt sind die File-Informationen in einem eigenen Chunk, dessen Name, und damit der Name des File, ebenso wie jeder andere Chunk per Hash aus dem Inhalt berechnet wird.

Die File-Chunks enthalten neben einigen Verwaltungsinformationen eine Liste der Namen aller Chunks, die zum File gehören. Abbildung 5.2 stellt diesen Aufbau dar. In diesem Entwurf von Wotan werden keinerlei Nutzdaten in der File-Struktur abgelegt, sondern immer in externen Chunks. Eine mögliche Verbesserung dieses Modelles wäre eine weitere Annäherung an die Struktur der I-Nodes des Unix-Dateisystemes (**?**, Seite 447 und 743). Da die

Chunks größer als übliche Disk-Blocks sind (1 MByte statt 1-4 KByte), würde es sich evtl. auch lohnen, die Daten von kleinen Dateien direkt in die File-Struktur zu schreiben.

Die Größe der Chunks ist aufgrund der wahrscheinlichen Hauptnutzung von Wotan für die Verteilung von Multimedia-Daten gewählt. Da bei Musik-, Bild- und vor allem Videodaten davon auszugehen ist, dass Dateien größer als 1 MByte sind, verringern große Blöcke den Overhead für die Übertragung über das Netzwerk. Für die Übertragung eines Chunks in Wotan ist jeweils eine Anfrage von mehreren 100 Byte notwendig und bedingt mehrere Pakete in beide Richtungen der TCP Verbindung. Darüber hinaus verringern die großen Pakete den genutzten Bereich des Namensraumes und reduzieren so die Wahrscheinlichkeit einer Namenskollision.

Die Infrastruktur von Wotan transportiert ausschließlich Chunks von einem Knoten zum nächsten. Sie hat kein Wissen darüber, welche Chunks zu einem File gehören. Auch den Umstand, dass es sich bei einen Chunk um einen File-Chunk handelt, muß die Infrastruktur nicht kennen.

Damit Anwendungsprogramme ein Speicherobjekt laden können, wird in der Wotan API über den Namen eines File auf das Speicherobjekt zugegriffen. Der Wotan Broker rekonstruiert das Speicherobjekt aus den Chunks, bevor es die Nutzdaten an die aufrufenden Applikation weitergibt.

5.4.2 Wotan Knoten

Wotan Knoten sind alle Computer, auf denen der Wotan Broker installiert ist und die an der Funktionalität des Wotan Systems teilhaben. Ihre minimale Funktion ist die Weiterleitung von Chunks von einem Knoten zum nächsten.

5.4.3 Identity Knoten

Identity Knoten sind Wotan Knoten, auf denen eine oder mehrere Benutzer-Identitäten eingetragen und aktiviert sind. Um eine geladene Identität zu aktivieren, muß der Benutzer seinen privaten und geheimen Schlüssel freischalten. Dies geschieht üblicherweise durch die Eingabe einer PIN oder ein vergleichbares Authentifikation-Verfahren. Die Details sind aber implementationsabhängig.

Nur auf ID-Knoten können Benutzer oder ihre Anwendungen auf die Daten zugreifen, die von Wotan verwaltet werden. Die Identity-Knoten verwalten darüber hinaus die Schlüssel, die für einen Zugriff auf die Chunks notwendig sind.

Aktive ID-Knoten agieren außerdem als Watcher für die Rückwärts-Notifikation. Um aktuelle Informationen über die Speicherorte der Chunks zu erhalten, synchronisieren ID-Knoten beim Betreten des Wotan Netzwerkes oder nach Beendigung einer Netzwerk-Partitionierung ihre Chunk-Location-Listen mit Hilfe des Identity Update Protokolls (siehe Abschnitt 5.5.4). Damit ist auch gewährleistet, dass alle Nachrichten der Rückwärtsnotifikation für die Chunks

einer Identity empfangen werden, solange in jeder Netzwerk-Partition mindestens ein ID-Knoten dieser Identity aktiv ist.

5.4.4 Overlay Netzwerk

Wotan nutzt zwei separate DHT Overlay-Netzwerke (siehe Abschnitt 3.1.1) für unterschiedliche Protokolle.

Das erste verwendet den Namen der Chunks als Schlüssel. Es dient dazu, Knoten zu finden, die an Informationen über den Speicherort von Chunks interessiert sind. Mit diesen Informationen können Identity-Knoten Listen der Speicherorte von Chunks pflegen. Auf diesen Mechanismus der Rückwärts-Notifikation wird in Abschnitt 5.3.1) genauer eingegangen.

Das andere Overlay-Netzwerk dient der Verwaltung der Identity-Knoten und nutzt den Hash der Identität als Schlüssel. Es erlaubt eine Kontaktaufnahme zwischen Identity-Knoten einer Identität, so dass diese ihre eigenen Informationen abgleichen können.

5.4.5 Clearing House oder Trusted Third Party (TTP)

Mit der Entgegennahme von Chunks durch Knoten B von Knoten A, leistet Knoten B einen Dienst für A und übernimmt die Verantwortung für die Weiterleitung des Chunks. Knoten die an einer Transaktion beteiligt sind, müssen abschätzen können, wieviel Vertrauen sie in andere Knoten legen können, die übertragene Aufgabe zu erledigen. Damit nicht jeder Knoten eine vollständige Transaktions-Historie des jeweils anderen Knotens braucht, existiert eine Clearing-Stelle, die aus Abrechnungs-Informationen einen Vertrauens-Index der Knoten berechnet.

Die Knoten benötigen keinen kontinuierlichen Internet-Zugang zur Clearing Stelle, sondern nur eine Möglichkeit, die Informationen über den Vertrauens-Index von der Clearingstelle zu erhalten.

In Abschnitt 5.9 wird auf die Abrechnung von Leistung und Risiko genauer eingegangen.

5.4.6 Wotan Communities

Die Communities sind ein rein organisatorisches Konstrukt und umfassen alle Wotan Knoten, welche die gleichen Parameter für die Chunk-Größe, die verwendeten Hash- und Verschlüsselungs-Algorithmen verwenden. Vor allem aber hat jede Community genau eine Trusted Third Party, um die erbrachten Leistungen und eingelösten Risiken abzurechnen.

5.5 Protokolle

In Wotan werden eine Reihe von separaten Protokollen für unterschiedliche Aufgaben eingesetzt. Abbildung 5.3 stellt diese im Überblick dar. Zwei dieser Prokotolle, Identity Discovery (Abschnitt 5.5.3) und Chunk Notification Broadcast (Abschnitt 5.5.2) werden wie erwähnt über ein DHT-Overlay-Netzwerk (siehe Abschnitt 3.1.1) abgewickelt. Alle anderen sind direkte TCP Point-to-Point Verbindungen.

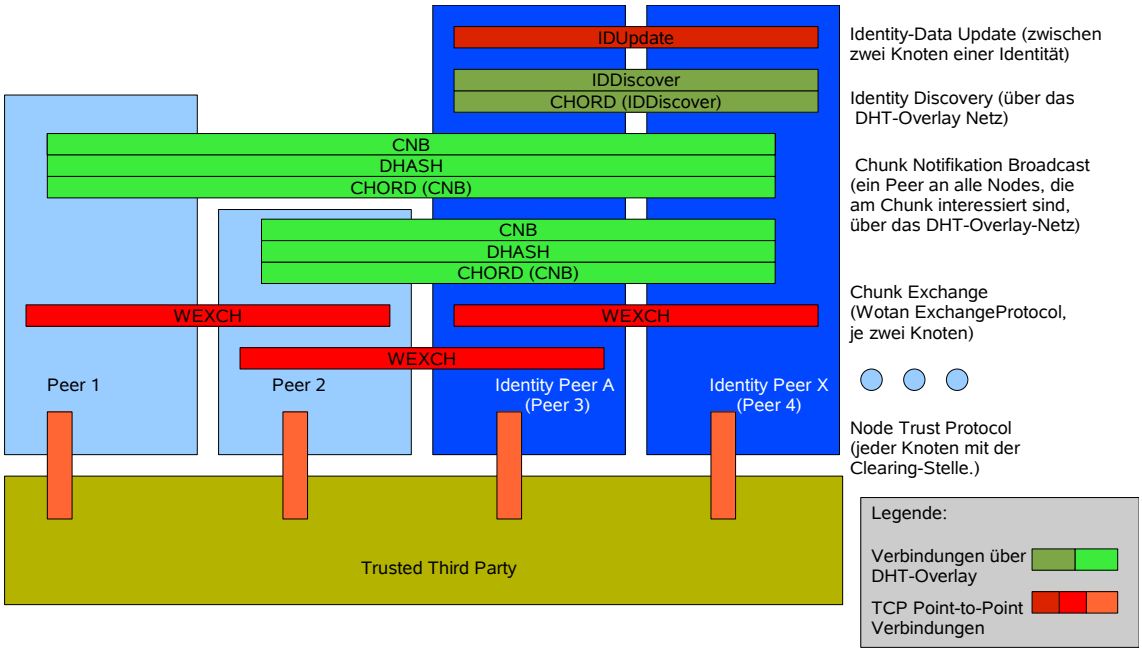

Abbildung 5.3: Übersicht über die unterschiedlichen Protokolle in Wotan

Alle Befehle, die direkt an die Knoten gehen, werden durch den gleichen Mechanismus wie beim Austausch von Chunks (siehe Abschnitt 5.8.2) durchgeführt. Das Wotan Exchange Protokoll (Siehe Anhang B) enthält darüber hinaus Kommandos für andere Funktionen die im Folgenden beschrieben sind.

5.5.1 Wotan Chunk Exchange Protocol (WEXCH)

Die Kommandos `SENDCH`,`QUERYCH` erlauben die Übertragung von Nutzdaten zwischen Knoten und stellen die Basis des Wotan Systems dar. Details zu der Entscheidung, ob und welche Chunks zwischen zwei Knoten übertragen werden sind in Abschnitt 5.8 beschrieben.

5.5.2 Chunk Notifikation Broadcast Protocol (CNB)

Das Chunk Notifikation Broadcast Protocol benachrichtigt die Knoten, die an Informationen über Speicherorte von Chunks interessiert sind, die Watcher. Wotan pflegt die Menge der Watcher-Listen (eine pro Chunk-Namen) mit Hilfe des DHash Protokolls (siehe Abschnitt 4.3.4.1 und **?**), welches wiederum auf der Chord DHT aufbaut. Das DHash liefert eine stabile Verteilung der Listeninformation zwischen den Knoten, so dass es einen ausgezeichneten Knoten gibt, an den sich die anderen Knoten wenden können. DHash verwendet Erasure Coding, um die Informationen über eine Liste auf mehrere Knoten zu verteilen, so dass der Verlust der Verbindung zu einem Knoten nicht zu einem Verlust von Daten führt.

Trifft ein Chunk auf einem Knoten A ein, so muß dieser die Watcher durch eine Chunk-Location Nachricht informieren. Dazu ermittelt der Knoten mit Hilfe von Chord die Adresse eines Knotens B, der für die Verwaltung des Watcher-List für den gegeben Chunk Namen zuständig ist. Dann kontaktiert er diesen Verwalter der Watcher-Liste mit dem Befehl `GETWATCH`.

Der Knoten B muß nun mit Hilfe des DHash Protokolles die Watcher-Liste rekonstruieren. Dazu rekombiniert er die Segmente der Liste mit Hilfe des Erasure Coding. Der Knoten kann die zusammengestellten Listen zu einem gewissen Maß zwischenspeichern, um die Effizienz der Anfrage zu steigern. Dann überträgt er die Liste an Knoten A.

Knoten A ruft nun auf jedem Knoten auf der Liste über das Wotan Protokoll den Befehl `CHUNKLOC` auf und meldet dem Watcher die eigene IP Adresse. Der Watcher kann daraufhin die eigene Liste der Speicherorte für den Chunk pflegen.

Dieses Protokoll ist nicht als Multicast implementiert, da jeweils nur ein sehr kleiner Teil der Wotan Knoten an Nachrichten über einen bestimmten Chunk interessiert sind, im Normalfall die Anzahl an Knoten, auf denen die Identity eines Benutzers installiert ist. Die Anzahl der Multicast Gruppen in IPv4 ist bei weitem zu klein, um für jeden Chunk eine Multicast Gruppe einzurichten. Würde man alle Nachrichten über die gleiche Gruppe verbreiten, wäre das Verhältnis zwischen gewünschten und empfangenen Nachrichten extrem schlecht. Darüber hinaus ändern sich mobile Umgebungen zu dynamisch, um ein stabiles Multicast in IPv4 realisieren zu können.

5.5.3 Identity Discovery Protokol

Betritt ein Identity-Knoten das Wotan System, so versucht er andere Kopien der Identität zu finden, um sich mit ihnen zu synchronisieren. Dieses Betreten von Wotan kann sowohl durch das Laden einer Identität auf einem Knoten, als auch durch ein Wiederbetreten nach einem Ausfall des Knotens oder einer Netzwerk-Partitionierung geschehen.

Die Adressen für die Identities werden mit Hilfe eines Lookups in einem eigenen Chord Overlay-Netzwerk realisiert. Die Knoten dieses Netzwerkes sind auf jedem Wotan Knoten

verteilt und Bestandteil des Wotan Basis-Systems. Der Lookup der Adresse geschieht gemäß folgendem Pseudo-Code:

```
adr = chord_lookup(id_hash)
id_adr = undef
solange id_adr undefiniert ist:
   adr = chord_get_successor(adr)
   id_adr = adr.getidnode(id_hash)
ende
```

Da ein einzelner Knoten mit den Informationen zu einer Identity jederzeit ausfallen kann, sind alle weiteren ID-Knoten dieser Identity als `successor()` in Chord organisiert. Dazu trägt sich jeder Identity-Knoten, der das System betritt, auf einem anderen Wotan Knoten ein, der von Chord bestimmt wird. Dies funktioniert gemäß folgendem Pseudo-Code, wobei `adr` durch den obigen Code, mit dem ersten Knoten, der die Adresse eines Identity-Knotens zur gegeben Identity enthält, vorbelegt ist:

```
solange id_adr != undef
   adr = chord_get_successor(adr)
   id_adr = adr.getidnode(id_hash)
ende
adr.setnode(self.ip_adr, id_hash)
```

Dabei sind `chord_get_successor` und `chor_lookup` Funktionen des Chord Protokolles und `getidnode` und `setidnode` Kommandos des Wotan Protokolles die auf dem Knoten `adr` aufgerufen werden.

5.5.4 Identity Update Protocol (IUP)

Hat ein Identity-Knoten mindestens einen anderen Knoten dieser Identität finden können, so beginnt er mit dem IUP, um die Informationen über die Speicherobjekte des Benutzers abzugleichen. Dabei sind mit Ausnahme des Protokoll-Kommandos (`IUQUERY`) alle Daten der Protokollnachrichten mit dem Symetric Master Key (siehe Abschnitt 5.6) dieser Identität verschlüsselt.

Der anfragende ID-Knoten erhält nun die Liste aller dem anderen ID-Knoten bekannten Files und ihrer Verschlüsselungs-Schlüssel. Der antwortende ID-Knoten kann nun die Rolle des Clients übernehmen (da er ja die IP-Adresse des anfragenden Knotens kennt) und selbst

RSA_{pub}, RSA_{priv} = genKeyPairRSA(2048)
SMK = random(256)
userID = $hash_{SHA-348}$(Benutzername,PKCS-8(RSA_{pub}, RSA_{priv}),SMK)

Tabelle 5.1: User Identity initialisieren

nach neuen Files fragen. Die Implementation des ursprünglich anfragenden Knotens kann den Datenaustausch optimieren und nur die Files melden, die dem ursprünglichen Server nicht bekannt waren. Aber dies ist nicht notwendig und eine optionale Funktionalität des Brokers.

Ein Knoten kann auch als Identity-Knoten an Wotan teilnehmen, ohne das IUP durchzuführen. Dies ist vor allem bei Geräten sinnvoll, die nur als Daten-Generatoren (z.b. Kameras) auftreten. Auf diesen Knoten muß jedoch die Identität des Benutzers geladen sein, um Informationen mit den notwendigen Schlüsseln zu verschlüsseln. Diese Knoten beantworten auch nicht die IUP-Anfragen von anderen Knoten, da davon ausgegangen werden kann, dass die Geräte nur aktiviert sind, wenn sie Speicherobjekte ablegen möchten. Um den Overhead zu verringern sollten sich derartig spezialisierte Identity-Knoten auch nicht in die Liste der ID-Knoten für das Identity Discovery Protocol eintragen.

5.5.5 Node Trust Protocol (NTP)

Um die Verläßlichkeit von Knoten bewerten zu können, stellt das NTP eine Bewertungskala zur Verfügung. Die Art, mit der Broker mit dieser Metrik umgehen, ist nicht Bestandteil dieser Arbeit. Wotan stellt lediglich eine Messgröße zur Verfügung, mit der ein Broker beurteilen kann, ob er mit der Anzahl der Kopien, die er versandt hat, die notwendige Sicherheit hat, die Daten des Anwenders sicher übertragen zu haben.

Die Messgröße im NTP ist das Elmonit (Electronic Monetary Unit). Der Wert, der einem Elmonit zugemessen wird, kann von Community zu Community unterschiedlich sein.

Die notwendigen Nachrichten für die Abwicklung des NTP zwischen den Knoten sind Bestandteil des WEXCH. Für die Kommunikation zwischen den Knoten und der Clearingstelle senden die Knoten ihre Receipts per HTTP an eine bestimmte URL auf der Clearingstelle und erhalten als Antwort Attribut-Zertifikate, die mit einer bestimmten Laufzeit das Vertrauen in den Knoten als Elmonit-Wert angeben.

5.6 Initialisierung

Vor der Verwendung von Wotan muß sich jeder Benutzer eine Identität erzeugen. Der Hauptzweck dieser Identität ist es, einen eindeutigen Identifier für die Verwendung im Chord DHT (Abschnitt 3.1) zu generieren.

Dafür erzeugt der Anwender ein RSA Schlüsselpaar. Bei der Wahl der Schlüssellänge ist Abschnitt 5.6.1 zu beachten. Eine Länge von mindestens 2048 Bit sollte von jeder Implementation unterstützt werden.

Zusätzlich erzeugt der Anwender einen Symetric Master Key (SMK). Dieser wird statt des asymetrischen Schlüssels verwendet, wenn dies möglich ist, um die Anzahl der asymetrischen Operationen zu reduzieren. Dieser Schlüssel sollte durch einen guten Zufallsgenerator (siehe Abschnitt 3.4.2) erzeugt werden, um eine unbeabsichtige Reduktion der effektiven Schlüssellänge zu vermeiden. Die Schlüssellänge sollte mindestens 256 Bit betragen.

Dazu wählt sich der Anwender nun einen Benutzernamen. Dieser ist in UTF-8 kodiert und kann von beliebiger Länge sein. Aus dem Schlüssel in PKCS#8 Format (**?**) und dem Benutzernamen wird nun ein Hash-Wert berechnet[5]. Dieser ist der Identifikator des Benutzers, der Wert $k_{IDDiscover}$ für die Verwendung in Chord.

Mit dem RSA Schlüsselpaar muß der Anwender nun einen Zertifikatsrequest erzeugen und an die Clearing-Stelle (Abschnitt 5.4.5) senden. Diese erstellt daraus ein digitales Zertifikat im x.509 (**?**) Format, das mit dem privaten Schlüssel der Clearing-Stelle signiert ist und schickt es an den Anwender zurück.

Diese Informationen (Privater Schlüssel, Zertifikat, Schlüsselpaar, DHT-ID) müssen auf jeden Knoten kopiert werden, von dem aus der Anwender Wotan verwenden möchte. Ein Knoten kann am Wotan Netzwerk teilnehmen, auch ohne dass ein Anwender seine Identität darauf abgelegt hat, kann dann aber keine neuen Speicherobjekte anlegen, sondern nur existierende verwalten und weiterleiten.

5.6.1 Schlüssellängen

Bei der Wahl der Schlüssellängen für alle kryptographischen Operationen, ebenso wie der Anzahl der Runden der kryptographischen Hashes in Wotan ist zu beachten, dass der Schutz durch die Verschlüsselung so lange wirksam sein muß wie ein Datenobjekt von Interesse ist. Anders als ein digitales Zertifikat läuft die Gültigkeit eines Datenobjektes nicht nach einer bestimmten Zeit ab, sondern muß geschützt werden, bis das Interesse einer dritten Partei an den enthaltenen Informationen erlischt.

Durch die Architektur von Wotan ist es nicht möglich, ein Datenobjekt, das einmal in Wotan abgelegt wurde, verläßlich zu löschen. Zwar gibt es eine Nachricht an beteiligte Knoten, dass ein Objekt nicht mehr von Interesse ist, aber die Natur von Wotan macht es unmöglich, die beteiligten Knoten daran zu hindern, diese Löschanweisung zu ignorieren.

Eine Berechnung der benötigten Schlüssellänge wird auf der Webseite (**?**) ausführlich diskutiert. Hier finden sich auch Hilfen für die Berechnung der Schlüssellängen, die notwendig sind, um ein Objekt bis zu einem bestimmten Termin sicher zu verwahren.

[5] Das Wotan Protokoll schreibt die Verwendung des SHA-384 Hash vor

5.7 Finden lokaler Knoten

Wie jedes andere Peer-to-Peer System auch (siehe Abschnitt 3.2) benötigen die Wotan Knoten Kenntnis über mindestens einen anderen Knoten, um Verbindung mit dem Rest des Systems aufnehmen zu können. Während in anderen Systemen entweder einige ausgezeichnete Knoten allgemein bekannt sind (z.B. Napster) oder einen Teil der Informationen zu der Datei bilden, auf die zugegriffen werden soll (z.B. BitTorrent), so gibt es in Wotan keinerlei derartige ausgezeichnete Knoten.

Andere Wotan Knoten müssen in Abhängigkeit von der verwendeten Netzwerk-Umgebung gefunden werden. In Bluetooth Umgebungen z.B bietet sich das Service Discovery Protocol (SDP, **?**, Seite 115ff) an, da es bei der ersten Kontaktaufnahme von zwei Bluetooth-Geräten ohne zusätzlichen Netzwerk-Overhead alle angebotenen Dienste übermittelt.In TCP/IP Umgebungen bietet sich das Service Location Protocol (**?**) an. Es kann innerhalb einer Broadcast-Domäne Anbieter von Diensten identifizieren.

Das Design von Wotan abstrahiert von der konkreten Technologie, die zum Finden von anderen Wotan Knoten verwendet wird. Sowohl der Aufbau der DHT-Routing Tabellen als auch der Austausch von Chunks (siehe Abschnitt 5.8.2) geht von der Annahme aus, das ein implementations-spezifischer Discovery-Mechanismus kontinuierlich neue Knoten findet, mit denen der Broker den eigenen Wissensstand abgleichen kann.

5.8 Chunk Verteilung

Die Verteilung von Chunks zwischen den Knoten ist die Kernfunktionalität innerhalb von Wotan , da hiermit die eigentliche Weiterleitung von Speicherobjekten geleistet wird.

Dabei folgt die Verteilung einem Store-and-Forward Grundprinzip wie es vom SMTP (**?**) bekannt ist. Ein Knoten überträgt einen Chunk an einen anderen Knoten, und dieser kann nun entscheiden ob er genug Speicherplatz hat, um den Chunk dauerhaft zu speichern, oder ob der Chunk bei nächster Gelegenheit wieder abgegeben wird (z.b. wenn der Besitzer das Gerät in die Reichweite eines WLANs trägt).

5.8.1 Policy Engine

Die Verteilung der Chunks zwischen den Knoten wird anhand lokaler Entscheidungen jedes Knotens getroffen. Dazu verfügt der Broker über eine Komponente, die Policy-Engine, die es ihm erlaubt, bei einem Kontakt zu einem anderen Knoten auf dem lokal vorhandene Chunks weiterzuleiten, sowie den kontaktierten Knoten nach gesuchten Chunks zu befragen.

Es gibt keine Vorgabe an eine Implementation des Brokers, wie die Policy-Engine die Chunks weiterzuleiten hat. Die Anforderungen an diese Policy ergeben sich aus der Notwendigkeit, innerhalb des Abrechnungsmodelles einer Wotan Community (Abschnitt 5.4.6) zu

funktionieren und eine möglichst hohe Rate an erfolgreich zur Verfügung gestellten Chunks für den Anwender zu realisieren.

In der partiellen Test-Implementation des Brokers wurde eine Policy-Engine realisiert, welche die Übertragung jedes Chunk im lokalen Cache an jeden Knoten, mit Kontakt aufgenommen wird, veranlasst. Dies sichert eine vollständige Übertragung aller Daten, stellt aber die oberste Grenze der Nutzung von Netzwerk- und Speicher-Ressourcen dar.

Eine Detailierung der Policy-Engine ist nicht Bestandteil dieser Arbeit, da sie direkt von einem vollständigen Abrechnungs-Modell abhängt.

5.8.2 Chunk Austausch

Die Policy-Engine nimmt bei einem Kontakt mit einem anderen Knoten immer die Rolle eines Clients ein, der die Übertragung oder Anfrage von Informationen initiiert. Damit aber beide Knoten gleichberechtigt die Funktionalität ihrer Policy-Engines ausführen können, kontaktiert ein Knoten A, der in der Rolle als Server von einem Knoten B als Client kontaktiert wurde, sofort auch als Client den Knoten B als Server. Abbildung 5.4 veranschaulicht diesen Ablauf.

Um Chunks zwischen den Knoten austauschen zu können, existiert das Wotan Chunk Exchange Protocol (siehe Abschnitt 5.5.1). Es definiert Protokoll-Befehle, mit denen die Knoten Chunks und Informationen über Chunks austauschen können.

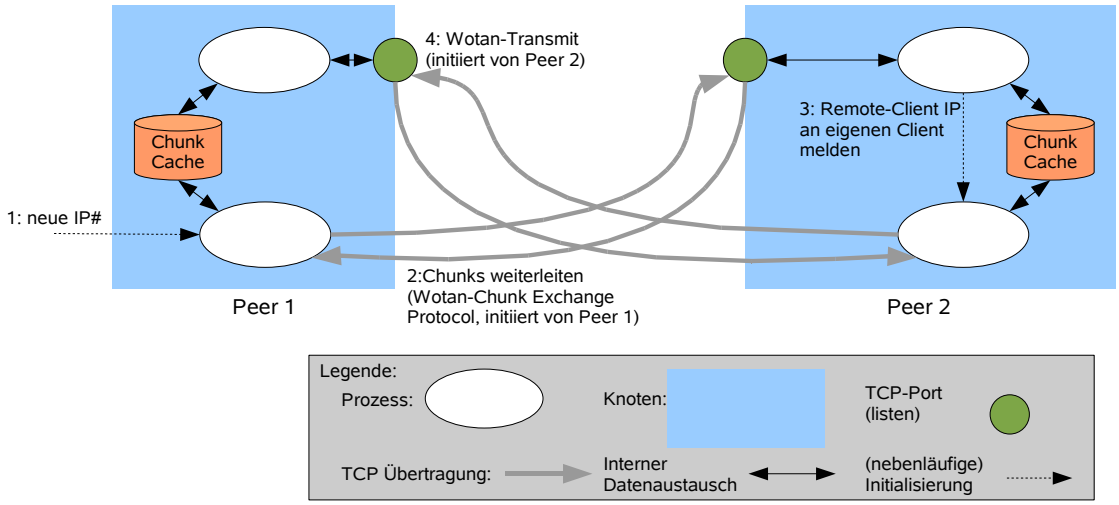

Abbildung 5.4: Austausch von Chunks zwischen den Peers.

5.8.3 Erasure Coding oder Replikation

Anders als CFS (**?**) benutzt Wotan Replikation statt Erasure Coding, um die Robustheit der Speicherung zu gewährleisten.

Wie in (**?**) genauer beschrieben, hat Erasure Coding gegenüber Replikation bei gleichem Platzbedarf eine deutlich höhere Stabilität gegen Datenverlust durch Ausfall von Speicherknoten.

Aber die Untersuchungen gehen davon aus, dass zu jeden Zeitpunkt genügend Knoten erreichbar sind, um die Fragmente auch auf unterschiedlichen Knoten ablegen zu können. Diese Annahme ist korrekt für ein im Internet verteiltes System, in dem die einzelnen Knoten den Großteil der Zeit direkt ereicht werden können. In einem mobilen Szenario, wo Netzwerk-Partitionierung als der Normalfall angenommen werden muß, kann davon nicht ausgegangen werden

In Wotan wird jedem Objekt durch den Ersteller ein Wert im Maßstab des Nano-Payments (siehe Abschnitt 5.9) zugewiesen. Dadurch ist es möglich, die Replikation in Abhängigkeit vom Wert des Speicherobjektes durch die Policy-Engine des Brokers regeln zu lassen. Bei Objekten, die einzeln nur geringen Wert haben, kann auf Replikation ganz verzichtet werden.

Für die Verwaltung der Chunk-Watcher Tabellen verwendet Wotan Erasure Coding, da es durch DHash zur Verfügung gestellt wird.

5.9 Nano Payment

Dieses Protokoll ist noch nicht bis in die letzten Details ausgearbeitet und soll hier nur dazu dienen die grundsätzliche Idee verständlich zu machen.

Um die Tätigkeit der Knoten als Dienstleistung abrechenbar zu machen und auch eine Messlatte für Fehlverhalten innerhalb der kooperativen Umgebung des Wotan Systems zu haben, existiert ein Protokoll für den sicheren Nachweis, dass Chunks weitergeleitet wurden oder dass die Verantwortung für die dauerhafte Speicherung von Daten übernommen wurde.

Die Übertragung der Nachrichten ist Teil des WEXCH. Im Folgenden wird als Empfänger die Maschine bezeichnet, auf dem der Chunk gespeichert werden soll, als Sender jene Maschine, die einen Chunk zum Speichern abgeben möchte .

1. Der Sender gibt mit dem Aufruf, einen Chunk senden zu wollen einen Wert des Chunks an (in Elmonit).

2. Der Empfänger gibt einen *Tender* ab: Er sagt was er für das Weiterleiten des Chunks haben möchte. Der Tender ist wie folgt aufgebaut: $T = encrypt_{priv_{empfnger}}(CN, PreisinElmonit)$

3. Der Sender kann das Angebot ablehnen, und der Protokollschritt ist beendet. (Der Sender kann nun mit einem anderen Wert oder einem anderen Chunk den Schritt wiederholen, aber dann muß er bei 1. beginnen.

Akzeptiert der Sender das Angebot, schickt er die Daten und einen Receipt, mit dem er bestätigt, den Auftrag erteilt zu haben. Der Receipt ist folgt aufgebaut: $R = encrypt_{priv_{sender}}(T)$

CN ist der Name des Chunk, also ein Hash des Inhaltes, T der Tender des Empfängers.

Beide Parteien leiten Tender und Receipt direkt oder indirekt an die TTP, weiter wenn die Dienstleitung erbracht wurde bzw. wenn das Risiko an eine andere Partei abgegeben wurde. Die TTP berechnet daraus ein Guthaben des Knotens (in Elmonit) und erstellt ein Attribut-Zertifikat mit diesem Wert, einem Datum und einer relativ kurzen Gültigkeitsdauer (ca. einige Tage, evtl. nur einige Stunden). Dieses Zertifikat tauschen Sender und Empfänger bei einem Verbindungsaufbau aus und können so die Reserven und die Vertrauenswürdigkeit des Kommunikatiospartners abschätzen (bzw. die Fähigkeit, für Fehlleistung einzustehen).

5.10 File Verwaltung

Die Verwaltung der Files und vor allem die Verbindung von für Menschen unleserlichen Hash-Werten mit verständlichen Informationen zu den Files sind weitere Funktionen, die mit Absicht nicht Bestandteil von Wotan sind. Stattdessen soll Wotan als Middleware dienen, auf deren Basis dem Anwender Programme die Speicherobjekte in einer dem Objekt angemessenen Art präsentieren. Beispiele wären Thumbnails für Bilder oder eine Verlinkung auf Einträge in einer Filmdatenbank für Filme.

Damit Anwendungsprogramme auf die in Wotan gespeicherten Daten zugreifen können, müssen sie zwei Informationen besitzen: den Namen des Files (einen Hashwert) sowie den geheimen Schlüssel, mit dem die Daten im File verschlüsselt wurden.

Beide Informationen müssen auch weitergegeben werden, wenn ein File an einen anderen Anwender weitergegeben wird. Eine Revokation der Zugriffsrechte ist damit natürlich nicht mehr möglich, sobald der Schlüssel erst einmal weitergegeben wurde. Aus Sicht des Kryptographen gilt eine Datei als kompromittiert, sobald ein Zugriff für einen gewissen Zeitraum möglich war (unabhängig von der Länge des Zeitraumes). Daher erscheint mir die fehlende Revokation nicht als wesentlicher Nachteil.

Im WEXCH existiert die Möglichkeit, dass ein Knoten einem anderen mitteilt, welcher Chunk ein File-Chunk ist. Dies ist aber keine Funktion der File-Verwaltung, sondern lediglich eine Möglichkeit der Optimierung der Chunk-Zugriffe. Unter bestimmten Umständen kann die Policy-Engine Information über die Zusammgehörigkeit von Chunks nutzen um ein Pre-Fetching zu implementieren. Die Implementation dieser Funktion ist optional.

Anfragende ID-Node	**Antwortende ID-Node**
Q= (“filekeyquery”,Chunk-ID des Files) Q an alle ID-Nodes von User A versenden	
	R = $encSMK_{userA}$(“filekeyreply”,Chunk-ID des Files, File Key) R an Absender von Q per TCP schicken.

Tabelle 5.2: Austausch von File-Encryption Schlüsseln zwischen ID-Nodes von User A

5.11 Chunk Retrieval

Wie in Abschnitt 5.4.3 beschrieben, kann nur über einen ID-Knoten auf die Speicherobjekte zugegriffen werden. Für einen Zugriff auf ein File kontaktiert der ID-Knoten die Speichorte der benötigten Chunks. Dabei befragt er jene Knoten, die sich als Speicherort für die Chunks über die Rückwärts-Notifikation gemeldet haben, beginnend mit dem spätestem Eintrag. Sobald der Chunk auf einem Knoten gefunden wird, kann er von dort geladen werden.

Für die ID-Knoten kann es sinnvoll sein, nur die letzten n Nachrichten aufzubewahren, um so den benötigten Speicherbedarf zu minimieren. Eine weitere Optimierung des Protokolles wäre es, eine Lösch-Nachricht an den speichernden Knoten senden zu können, so dass dieser einen Chunk löschen kann, wenn er auf ausreichend vielen anderen Knoten gespeichert ist. Eine weitere mögliche Erweiterung wäre eine Nachricht des speichernden Knotens an die Watcher, dass er die Speicherung eines Chunks nun aufgibt, um so die Aktualität der Informationen über Chunk-Speicherorte bei den Watchern zu verbessern.

5.12 Daten Verschlüsselung

Alle Speicherobjekte in Wotan werden immer verschlüsselt und erst auf einem Knoten, auf dem die Identität eines Benutzers installiert ist, wieder entschlüsselt. Die dazu notwendigen geheimen Schlüssel werden als Teil der Identitätsinformationen verwaltet.

Erhält ein Knoten eine Nachricht der Rückwärts-Notifikation (siehe Abschnitt 5.3.1) für eine Datei, dessen Schlüssel, K_{F_n}, dieser Knoten gerade nicht verwaltet, so schickt er eine Nachricht über das DHT-System an alle Identity-Knoten. Enthält ein Knoten den entsprechenden Schlüssel, so wird dieser mit dem SMK verschlüsselt und an den fragenden Knoten zurück gesandt (vergl. Abschnitt 5.2).

Die Anfrage selbst ist nicht verschlüsselt, da ohnehin die Eigentümer ID jedes Chunks ermittelbar ist und auch bekannt sein muß, damit die Rückwärts-Identifikation (siehe Abschnitt 5.3.1) funktionieren kann.

Sollte kein Knoten den Schlüssel für die Datei kennen, so wird dieser als unbekannt markiert und wird erst bei Verwendung der Datei durch den Benutzer wieder nachgefragt.

Sollte auch dann der Schlüssel nicht zu ermitteln sein, dann wird dem Benutzer ein Fehler gemeldet.

5.13 Verlauf der Datenspeicherung

Im Folgenden werden anhand der Dschungel-Patrouille (siehe Abschnitt 2.1) beispielhaft die Schritte des Wotan Systems dargestellt.

Alle Kameras und Mobiltelefone mit Kamera, die als Erzeuger der Daten dienen sollen, müssen als ID-Knoten mit den Identitäten ihrer Besitzer (oder momentanen Benutzer) geladen und aktiviert werden. Wie dies genau geht, hängt vom einzelnen Gerät ab.

Nun können die Anwender Fotos schießen. Die Fotos werden dem lokalen Wotan Broker auf den Kameras übergeben. Dieser läßt sich vom Zufallszahlengenerator einen Schlüssel für die Verschlüsselung erzeugen und verschlüsselt damit das Speicherobjekt, teilt es in Chunks auf und schreibt die Namen der Chunks in ein File. Der Broker trägt den Schlüssel in die lokale Liste der Speicherobjekte (also der Fotos), zusammen mit dem Namen des gerade angelegten File ein.

Nun konsultiert der Broker die Liste, der ihm bekannten, anderen Wotan Knoten. Er versucht diese zu kontaktieren. Ist keiner der Knoten aus der momentanen Netzwerkumgebung heraus erreichbar, versucht er mit Hilfe seiner Anbindung an das lokale Netzwerk (IP über WLAN, Bluetooth, etc.) lokale Wotan Knoten zu finden. Diese Knoten werden als nächstes kontaktiert. Der Broker versucht so lange mit anderen Knoten Kontakt aufzunehmen, bis ihm dies gelingt. Einstellung zum Energiesparen können die Häufigkeit seiner Versuche einschränken.

Gelingt dem lokalen Broker die Kontaktaufnahme mit einem andern Wotan Knoten, so überträgt der Broker die Chunks des Speicherobjektes. Diesen Schritt wiederholt er so häufig bis die lokale Policy-Engine sicher ist, dass, für den Wert des Bildes (dieser wird vom Benutzer gesetzt) ausreichend viele Kopien an andere Knoten verteilt wurden. Danach löscht der lokale Broker die Chunks des Speicherobjektes, und behält nur noch Namen der Datei und Schlüssel, zusammen mit einigen geeigneten Metadaten für die Darstellung in einem User-Interface (z.B. ein kleines Thumbprint).

Der nächste Knoten, der einen der Chunks übernommen hat, kann nun optional versuchen die Watcher für diesen Chunk zu benachrichtigen. Dies macht nur Sinn, wenn der Knoten einen stabilen Zugang zum Internet hat. Sonst würden Anfragen für den Chunk, den ein ID-Knoten stellen könnte u. U. nicht beantwortet, weil der Kontakt des speichernden Knotens zum Internet gerade abgerissen ist.

Der Knoten, welcher den Chunk übernommen hat, kann aber bei einer Verbindung zu einem oder mehreren weiteren Knoten, die Chunks weiterleiten, und sich so der Verantwortung, die Chunks sicher zuverwahren, wieder entledigen. Das Nano-Payment-Modell dient dazu diese Dienstleistung abzurechnen. Damit wandert der Chunk von einem Knoten zum

nächsten. Diese Wanderung muß nicht unbedingt enden, sondern kann theoretisch kontinuierlich weitergehen.

Da ein sinnvolles Abrechnungmodell aber die Dienstleistung der Speicherung nach der Frist, die ein Chunk auf einem Knoten abgelegt war, bezahlen müsste, gibt es einige Knoten, die über ausreichende Stabilität der Netzwerk-Anbindung, ein Backup-Konzept für die Massenspeicher und andere Massnahmen einer qualitativ hochwertigen Rechenzentrums-Dienstleistung verfügen. Für diese Knoten ist das Risiko der Wert-Sicherung der Chunks vertretbar, und sie können einen Mehrwert durch die Erbringung der Speicherleistung erwirtschaften. Deswegen reichen sie die Chunks die sie erhalten nicht weiter, sondern legen sie dauerhaft auf den lokalen Massenspeichern ab. Natürlich benachrichtigen sie, die Watcher wenn ein Chunk auf dem Knoten eintrifft (siehe Abschnitt 5.5.2). Dieser Schritt ist die Rückwärts-Notifikation

Trifft nun der Benutzer der Kamera wieder zuhause ein, so überträgt er die Informationen über seine Bilder auf seinen privaten PC. Dies kann per Bluetooth oder per USB-Kabel geschehen. Ohne diese manuelle Übertragung aber kann der eigene PC nicht auf die Bild-Daten zugreifen, da nur hier die kryptographischen Schlüssel für die Bilder abgelegt sind. Natürlich könnte eine Kamera auch schon “vom Felde” aus die Informationen über die Bilder an andere Anwendungen verschicken, z.B. via GPRS.

Ist der PC, auf dem ein Wotan Broker läuft, auf dem auch die Identität des Benutzers geladen ist, eingeschaltet, so kontaktiert er die anderen ID-Knoten des Benutzers, z.B sein Notebook, oder seinen PC am Arbeitsplatz und gleicht die Informationen über die bekannten Files und Schlüssel mit ihnen mit Hilfe des IUP (siehe Abschnitt 5.5.4) a. Danach kann der Anwender von jedem seine Computer aus auf alle seine Daten zugreifen.

Um Files an einen anderen Anwender weiter zugeben, braucht der Erzeuger nur Namen des Files und den Schlüssel zu übertragen. Dannach kann auch der Empfänger von jedem *seiner* Computer aus auf die Daten zugreifen.

Um auf die Speicherobjekte nun wieder zugreifen zu können, lädt der Wotan Broker auf dem dafür benutzen Computer die Chunks dieses Files. Den Speicherort der Chunks erhält er aus der File Info Table, welche die ID-Knoten mit Hilfe der Rückwärtsnotifikation pflegen. Die Chunks werden mit dem Schlüssel aus der gleichen Tabelle entschlüsselt und dem Anwender (oder seiner Software) im Klartext und als kontinuierliche Datei zur Verfügung gestellt.

Kapitel 6

Fazit und Ausblick

Wie wohl bei den meisten akademischen Arbeiten stellt man zum Ende fest, das man vieles besser anders gemacht hätte, denn nun hat man endlich die notwendige Erkenntnis, das es bessere Wege gibt. Das folgende Kapitel detailiert welche Anforderungen vom Systementwurf erfolgreich erfüllt werden können, und wo bei späteren Systemen noch Raum für Verbesserungen ist.

6.1 Architektur und Protokolle

Wotan erfüllt funktional die Anforderungen nach eine Skalierbarkeit sowohl in Hinblick auf die Anzahl der beteiligten Knoten ebenso wie die Gesamtspeicherkapazität durch die Elemination eines zentralen Knotens. Ein Schreiben von Daten in das System hinein ist entgegen vieler andere Speichersysteme auf P2P Basis möglich. Ebenso erlaubt der Store-And-Forward Ansatz eine Verwendung in Umgebungen mit instabiler Netzwerk-Anbindung. Auch die Ende-zu-Ende Verschlüsselung wird durch die durchgängige Verschlüsselung aller Daten nicht nur möglich, sondern ist integraler Bestandteil und kann auch nicht durch den Anwender deaktiviert werden.

Eine Discovery aller Knoten ist möglich, verläßt sich aber auf die Fähigkeiten externer Protokolle, die für diesen Zweck eingesetzt werden.

6.1.1 Protokolle

Bei der sequentiellen Vervollständigung der Funktionen von Wotan wuchs ebenso sequentiell die Anzahl der parallel ablaufenden Protokolle. Damit hat das Wotan Gesamtsystem eine sehr hohe Komplexität, die zu einem nicht unerheblichen Protokol-Overhead führt. Die notwendigen Nachrichten für die Pflege der zwei separate DHTs und die anderen Protokollelemente erfordern einen Overhead der dem vorgesehenen Einsatz in mobilen Umgebungen entgegensteht.

6.1.2 Stabilität gegen interne Angreifer

Die Pflege der Liste von Watchern läßt sich ohne weiteres durch einen Angreifer manipulieren und selbst wenn ein Löschbefehl nicht zum Einsatz kommt, mit einer Unmenge von falschen Einträgen zu einer Quelle für einen `Distributed Denial of Service` (DDoS) Angriff werden, wenn alle Knoten Update-Nachrichten an einen Computer schicken, der gar nicht Bestandteil des Wotan Systems ist. Selbst wenn der Aufbau einer TCP-Verbindung aufgrund eines geschlossenen Ports nicht stattfindet, so können die Initierungs-Pakete schon eine Netzwerkverbindung überlasten.

Ein Einsatz von starker Authentifizierung würde den Problematischen Einsatz von asymetrischer Kryptographie nur verschlechtern. Die notwendigen Rechenleistungs-Resourcen würden weiter wachsen. Außerdem müßte das Problem der Distribution von Zertifikaten an jeden Knoten gelößt werden, nicht nur für die Identity-Knoten wie es im momentanen Vorschlag der Fall ist.

6.1.3 Stabilität gegen verlorene Nachrichten

Das Design des IDDiscovery-Protokolles erfordert das mindest ein Knoten mit einer bestimmten Identity jederzeit aktiv ist, und auch nicht durch eine Netzwerk-Partitionierung von einem suchenden Knoten getrennt ist. Dies könnte durch einen statischen Dienste-Anbieter geleistet werden, der als Dienst für das Internet Identity-Knoten betreibt. Dies würde wieder einen herausragenden Punkt für einen Angriff liefern. Alternativ muß der Anwender seinen privaten PC ständig eingeschaltet lassen. Da dies aber die Möglichkeit eines Angriffes auf seinen, bei einem Heim-Anwender akzeptabel, nicht perfekt gepflegten Rechner wahrscheinlicher macht, ist diese Lösung auch nicht ohne eigene Probleme. Vor allem soll Wotan ja gerade die Trennung des Anwenders von ausgezeichneten Maschinen, hin zu einer Intelligenten Umgebung realisieren.

6.2 Verschlüsselung

Wotan setzt auf den massiven Einsatz von kryptographischen Verfahren. Neben einer notwendigen Überprüfung gibt es auch einige grundsätzliche Probleme die noch untersucht werden müssen.

6.2.1 Algorithmischer Durchbruch

Der Schutz der in Wotan verwallteten Verfahren hängt direkt von der Stärke der einegesetzten Kryptographischen Verfahren ab. Sollte bei den mathematischen Grundlagen der eingesetzten Verschlüsselungsverfahren ein grundlegender Durchbruch geschehen, so gibt

es keine Möglichkeit den Schutz der von Wotan verwalteten Dateien nachträglich zu verbessern. Anders als in anderen Systemen die Verschlüsselung nur für die Übertragung zwischen vertrauenswürdigen Parteien verwenden, legt Wotan daten ja auch bei einem potentiellen Angreifer ab. Es ist davon auszugehen das es irgendwo in der Welt einen Server gibt, der alle Daten, die jemals über das Wotan Netzwerk ausgetauscht wurden, aufgezeichnet hat und bei einem Durchbruch in der Kryptographie mit dem neuen algorithmischen Wissen alle Schlüssel bricht und dann alle Daten im Klartext lesen kann.

6.2.2 Einsatz asymetrischer Verschlüsselung

Entgegen der Anforderung, möglichst keine asymetrische Kryptographie zu nutzen, führt Wotan vor der Übertragung jedes Chunks zwischen zwei Knoten mindestens zweimal die `encrypt_private()` Funktion aus. Dieser Aufwand ist, wegen der langlaufen Rechenoperation, auf mobilen Geräten nur einer massiven Belastung des Akkus möglich. Die hiermit verbundene Verkürzung der Laufzeit des Gerätes steht einer weiten Verbreitung von Wotan im Wege.

Wenn sich in der Zukunft Hardware-Crypto Module, wie das Trusted Plattform Module (TPM, **?**) auch bei mobilen Geräten durchsetzen sollten, ist mit der damit verbundenen Hardware-Implementation der asymetrische Kryptographie ein Verbesserung bei Energiebedarf und Rechenzeit möglich. Damit könnte eine Hürde für die Verbreitung von Wotan reduziert werden. Aber hingegen der vollmundigen Versprechen der Designer des TPM seit 2001 über die Durchdringungsrate von Geräten mit dem TPM, wird es noch einige Jahre dauern bis TPMs auch in Mobiltelefonen als gegeben angenommen werden können.

6.3 Ausblick

Die Arbeit an Wotan hat mir sicherlich geholfen, den gesamten Bereich der verteilten, kooperierenden Datenspeicherung zu beleuchten. Und währen Wotan an sich kein erfolgversprechender Ansatz ist, so läßt sich evtl. die Rückwärts-Notifikation noch weiter entwickeln. Einige Ideen und Ansätze für weitere Untersuchungen sind im folgenden beleuchtet

6.3.1 Stabilität gegen Datenverlust

Die heute verwandte Replikation der Chunks hat ein fundamentales Problem, da ohne beachtlichen Aufwand nicht festgestellt werden kann, wieviele Replikation zu einem Zeitpunkt wirklich existieren.

Eine mögliche Lösung dieses Problems währe der Einsatz von Erasure Codes auch für Chunks. Dann könnte der Erzeuger n Fragmente anlegen und weiterleiten. Jeder Knoten, der Chunks, bzw. Chunk-Fragmente dauerhaft speichern möchte würde solange Fragmente

an andere Knoten weiterleiten, bis nur noch ein Fragment eines Chunks auf einem Knoten vorhanden währe. Knoten die Chunks nur weiterleiten, würden wie bisher alle Fragmente an einen anderen Knoten weiterleiten, sobald der Knoten die Möglichkeit dazu hat.

Das notwendige Protokoll zur dauerhaften Wartung der Fragmente, so das Verlust einzelner Fragmente nicht zu einem Verlust des Chunks führen kann, wenn weniger von m Fragmenten übrig sind (siehe Abschnitt 3.3), müsste allerdings noch entwickelt werden.

6.3.2 Ideen für ein Abrechnungsmodell

Während ein Abrechnungsmodell aus dieser Arbeit ausgeklammert ist, so haben sich doch während der Entwicklung von Wotan einige Anforderungen und Grundkonzepte, die ein solches Modell erfüllen muß, herausgestellt.

Ein möglicher Ansatz für ein Abrechnungsmodell, für das mit dem Nano-Payment auch schon der Grundstein gelegt ist, währe es, den Speicherobjekten einen Wert beizumessen. Dies ist auch für Geräte ohne Benutzerschnittstelle möglich. Die Speicherobjekte haben dann eben einen Einheitswert. Das grundsätzliche Modell geht dann davon aus, das jeder Knoten die Verantwortung für den Wert des Speicherobjektes übernimmt. Der Wert des Speicherobjektes, verbunden mit der Wahrscheinlichkeit der nicht erfolgreichen Weiterleitung, ergibt ein bestimmtes Risiko das mit Hilfe der Clearin-Stelle abgerechnet wird. Die Besitzer von Knoten im Wotan Netzwerk haben dann ein bestimmtes Guthaben für Risiko. Dies kann z.B. durch die Hinterlegung von realem Geld realisiert werden.

Knoten können dieses Risiko minimieren indem sie entweder die Daten des Speicherobjektes erfolgreich an andere Knoten weiterleiten. Oder, wenn sie über ausreichend Speicherkapazität verfügen, in dem sie dem Identity-Knoten des Besitzers den Speicherort melden und so eine nutzbare Dienstleistung für den Anwender leisten und dafür von der Clearing-Stelle eine gutschrift erhalten. Wenn der Anwender nun die gespeicherten Daten in Anspruch nimmt, in dem er sie von einem Knoten lädt, dann erhält der entsprechende Knoten eine weitere Gutschrift.

Anhang A

Glossar

E2EE End-to-End-Encryption, Ende-zu-Ende Verschlüsselung. Die verwalteten Daten werden auf dem Computer des Erzeugers verschlüsselt und während des gesamten Zeitraumes der Übertragung und Ablage in verschlüsselter Form bewahrt. Erst bei einem Zugriff eines Anwenders werden die Daten auf dem Computer des Anwenders wieder entschlüsselt.

DHT Distributed Hash Table, siehe Abschnitt 3.1

MAC Mandatory Access Controls, verpflichtende Zugriffs-/Berechtigungskontrolle. Der Anwender/Erzeuger eines Speicherobjektes kann nicht entscheiden welche Rechte er auf diese Objekte gewähren möchte. Eine zentrale Systemeinheit, gesteuert durch den Administrator legt diese Fest. Im Gegensatz zu den *Discretionary Access Controls*, freiwilligen Zugriffskontrollen, bei denen der Eigentümer der Daten entscheidet welche Rechte auf diese gewährt werden.

MANet Mobile Ad-Hoc Network. Netzwerke die durch das physikalische Zusammentreffen von mobilen Geräten entstehen und in der Regel über drahtlose Netzwerkverbindungen (Bluetook, WLAN, SensorNet) aufgebaut werden. Diese Netzwerke sind ihrer Natur nach nur von kurzer Dauer, da die beteiligten Knoten jederzeit wieder physikalisch entfernt werden können.

P2P Peer-2-Peer, siehe Abschnitt 3.2

SMK Symetric Master Key, symmetrischer Master Schlüssel, siehe Abschnitt 5.6

Knoten Ein Knoten im Sinne von Wotan ist ein Gerät, das über eine gewisse Rechenleistung verfügt, sowie über eine Netzschnittstelle, die einen direkten oder indirekten (über einen anderen Knoten) Zugang zum Internet ermöglicht. Ein Wotan Knoten verfügt darüber hinaus mindestens über eine Implementation des Wotan Brokers und kann damit mit anderen Knoten des Wotan Systems in Kontakt treten.

Der Begriff Computer wird mit Absicht nicht verwandt, weil gerade jene Geräte, die der Benutzer bemerkt und in die Hand nehmen kann, nicht wie klassische Computer aussehen. Statt dessen handelt es sich viel wahrscheinlicher um Geräte wie Mobiltelephone, Digitalkameras, Fernseher oder Stereoanlagen. Natürlich könenn auch Notebooks und PCs Knoten im Wotan -Netz darstellen, aben nicht nur sie.

Kontext Im Sinne von Wotan stellt die Menge aller Knoten, die über eine gemeinsame Schnittstelle miteinander kommunizieren können einen Kontext dar. Dabei ist es nicht relevant, ob alle Knoten zu einem bestimmten Zeitpunkt auch wirklich zu Kommunikation bereit und in der Lage sind. Lediglich der Umstand dass sie durch technische und organisatorische Massnahmen prinzipiell in der Lage sind, an einem Wotan Kontext teil zu nehmen, macht sie zu Mitgliedern eines Kontextes.

Neben der Möglichkeit miteinander zu kommunizieren, teilen die Mitglieder eines Kontextes auch die Werte für etwaige kontext-abhängige Konfigurationsparameter. Diese beinhalten die Auswahl an Hash- und Verschlüsselungsalgorithmen ebenso wie die dazugehörigen Schlüssel-Längen und Chunk-Größen.

File Jedes Speicherobjekt, das von einem Benutzer angelegt wird, egal ob es sich um Bilder, Texte oder Audio-Daten handelt, wird in Wotan absolut gleich als File gehandhabt. Wotan verwaltet keinerlei Informationen über die Art des Inhaltes von Files.

Chunk Jedes File wird abhängig von seiner Größe in eine Anzahl von Blöcke aufgeteilt. Diese werden in Wotan als Chunks bezeichnet und sind innerhalb eines Kontextes alle von gleicher Größe. Die Identifikation eines Chunks geschieht anhand eines Namens, der sich mittels eines kryptographischen Hashwertes aus dem Inhalt des Chunks ergibt.

Die Informationen über die Chunk-Zusammensetzung eines Files werden auch in einem Chunk abgelegt, das von Wotan wie jedes andere Chunk auch verwaltet wird. Der Name eines Files ergibt sich damit aus dem Namen des Chunks, der zu Verwaltung des Files erzeugt wurde. Ein Zugriff auf ein File ist nur mit der Kenntnis dieses Namens möglich.

Anhang B

Wotan Exchange Protocol

B.1 Einleitung

B.1.1 Protokollelemente

Das Protokoll sorgt in der Hauptsache für die Übetragung von Dateien (“Files”) die in Abschnitte (“Chunks”) aufgeteilt sind. Die Chunks werden durch das Ergebniss eines kryptographischen Digest auf ihren Inhalt identifiziert. Für jede Datei gibt es mindestens einen Kontroll-Chunk, der eine Liste der zur Datei gehöhrigen Chunks enthällt. Eine Datei kann im Prinzip aus beliebig vielen Chunks bestehen, ist in dieser Version des Protokols aber auf die Anzahl der Chunks begrenzt deren Identifikatoren (“Hashes”) in einem Chunk abgelegt werden können.

Jede Form von Metadaten, die für die Präsentation der Dateien für einen Menschen notwendig sind, z.B. ein Dateiname oder Typ des Inhaltes sind nicht Bestandteil dieses Standards.

B.2 Verbindungsaufbau

Durch einen Mechanismuß der nicht Bestandteil dieses Protokolles ist werden Gegenstellen identifiziert. Diese werden durch den Aufbau einer TCP Verbindung kontaktiert. Da jeder Knoten im System sowohl als Client als auch als Server im Socket-Modell auftritt wird eine kontaktierte Gegenstelle auch die Gegenverbindung aufbauen.

Der Mechanismuß zum identifizieren von Gegenstellen kann sowohl eine statische Liste von Teilnehmern einer Gruppe wie auch ein AdHoc-Discovery Mechanismuß wie UPnP, SDP oder Bluetooth-Discovery sein.

Danach handeln beide die zu übertragenen Daten aus. Das folgende Protokol ist aus

Sicht des Clients definiert. Beide Knoten agieren nach dem gegenseitigen Verbindungsaufbau simultan als Client und führen das Protokol durch.

Die Entscheidung ob Daten angenommen oder abgeschickt werden sollen entscheidet eine Policy-Komponente in jedem Teilnehmer eigenständig. Diese Policy-Komponente ist nicht Bestandteil dieses Protokolles.

Wenn eine Partei nicht zeitgerecht auf ein Kommando antwortet (Es kommt an beliebiger Stelle zu einem Timeout) wird die Verbindung abgebrochen kann erneut aufgebaut werden wenn die Gegenstelle noch erreichbar ist.

B.3 Kommandos

Alle Zeichen sind nach dem ANSII kodiert. Die Antwortkodes bestehen jeweils aus Dezimalzahlen die als String von drei Zeichen kodiert sind.

Die mit **Send:** markierten Nachrichten werden vom Client and den Server gesandt, die mit **Reply:** markierten Nachrichten vom Server zu Client.

B.3.1 HELLO

Das HELLO-Kommando initiert die Kommunikation zwischen zwei Knoten und dient dem Abgleich der Versionsnummer, so das ein Teilnehmer unterschiedliche, evtl. inkompatible Versionen des Standards unterstützen kann.

Send: HELLO <Protokoll Version: 1 byte>

Reply: <Antwortkode: 3 byte>

Die Protokoll Version kann folgende Werte haben:

Version	*Byte Wert*
0.9	1

Der Antwortkode kann folgende Werte annehmen:

Kode	*Mnemonik*	*Bedeutung*
200	TREADY	Der Server ist bereit für weitere Schritte im Protokol und unterstützt die angebene Versionsnummer
403	VERFAIL	Die im HELLO angegebene Versionnummer wird vom Server nicht unterstützt
501	NETFAIL	Ein nicht näher spezifizierter Fehler in der Übertragung ist aufgetreten.

B.3.2 Send Chunk (SENDCH)

Send: SENDCH <Chunk Länge: Integer Wert, MSB, 4 Byte> <Hash OID Länge: 1 byte> <Hash Algorithmus, DER-kodierte ASN.1 OID> <chunk hash> <Chunk Value>

Reply: <Antwortkode 3 byte> <Tender>

Wenn der Antwortkode des Servers “200” ist:
Send: <Chunk daten>

Wenn die Übertragung erfolgreich war:
Reply: <Antwortkode: TROK> <chunk hash> <Signature Algorithm ASN.1 OID> <Signatur Daten> <Reciept>

Sonst:
Reply: <Antwortkode> [<Länge der Fehlerbeschreibung: 1 Byte>] [<Fehlerbeschreibung, max 255 Byte>]

Die textuelle Fehlerbeschreibung sollten in Englisch gehalten sein.

Alg	*OID (ASN.1/DER)*	*Länge (Bit)*
SHA	1.3.14.3.2.18 (06:05:2B:0E:03:02:12)	20
SHA-1	1.3.14.3.2.26 (06:05:2B:0E:03:02:1A)	20
SHA-384	2.16.840.1.101.3.4.2.2 (06:09:60:86:48:01:65:03:04:02:02)	48
SHA-512	2.16.840.1.101.3.4.2.3 (06:09:60:86:48:01:65:03:04:02:03)	64

SHA-384 ist der bevorzugte Algorithmus. Die angegebenen OID Informationen stammen aus der Datei `dumpasn1.config` des dumpasn1 Programmes von Peter Gutmann.

Folgende Fehlermeldungen können vom Server zurückgegeben werden:

Kode	*Mnemonik*	*Bedeutung*
200	TREADY	Der Server ist bereit für weitere Schritte im Protokol und unterstützt die angebene Versionsnummer
201	TROK	Die Übertragung des Chunks war erfolgreich
301	HAVECH	Einen Chunk mit diesem Hash hat der Empfänger schon
302	NOTENDER	Der Server ist nicht bereit die Verantwortung für den Chunk zu übernehmen.
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
402	NOSPACE	Empfänger hat keinen Platz oder ist zumindest nicht bereit Platz zu opfern.
501	NETFAIL	Ein, nicht näher spezifizierter Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter lokaler Fehler ist aufgetreten

B.3.3 Retrieve Chunk (QUERYCH)

Ein Client fordert einen Chunk vom Server ab.

Send: QUERYCH <hash OID len: 1 byte> <hash algorithm ASN.1 OID> <chunk hash>

Reply: <reply code 3 byte> [<vom Fehlerkode abhängige Daten>]

Folgende Fehlerkodes können auftreten:

Kode	*Mnemonik*	*Bedeutung*
200	TREADY	Der Server ist bereit den Chunk zu übertragen. Die Nutzdaten folgen dem Fehlerkode.
302	NOHAVE	der Chunk mit diesem Hash ist dem Server nicht bekannt
303	NOHAVENOW	der Chunk ist momentan nicht verfügbar. Er wird gerade von einem anderne Server/Medium besorgt und der Client kann, wenn er möchte, in der Zwischenzeit etwas anderes tun.
304	REDIRECT	Der Server ist nicht bereit, berechtigt oder sonst in der Lage den anderen Server, von dem er weis das dieser den Chunk hat, zu kontaktieren, aber der Client kann selbst die Verbindung aufbauen und die Daten holen.
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
501	NETFAIL	Ein, nicht näher spezifizierter, Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter, lokaler Fehler ist aufgetreten

Wenn der Fehlerkode 200 beträgt sieht die Antwort des Servers wie folgt aus:

Reply: <reply code 3 byte> <chunk length: integer value 4 bytes, little-endian> <chunk data>

Wenn der Fehlerkode 304 beträgt sieht die Antwort des Servers wie folgt aus:

Reply: <reply code 3 byte> <Addresse (IP) des andern Servers auf dem sich die Daten befinden können. 4 Byte>

B.3.4 Set File (SFILE)

Eigentlich ist eine Benachrichtigung des Peers über die Eigenschaft eines Chunk als Datei-Index nicht notwendig. Ein Zugriff auf eine Datei ist nur möglich, wenn eine Anwendung den Hashwert und damit den Identifier eines Index-Chunks verwaltet. Somit muß ein Server nicht wissen um was es sich bei einem Chunk handelt.

Allerdings kann es hilfreich sein, um die Offline-Effektivität von Clients zu verbessern,

alle Chunks einer bekannten Datei auf den lokalen Puffer zu übertragen. Dies ist nur möglich wenn der lokale Broker die Identifikatoren für alle Chunks einer Datei kennt.

Der Befehl `SFILE` muß von einem Server nicht implementiert werden, solange eine Anfrage mit dem korrekten Fehlercode (510:NOIMPL) beantwortet wird.

Der Befehl überträgt keine Chunk-Daten sondern markiert einen Chunk nur als Datei-Index.

Send: SFILE <Hash OID len: 1 byte> <Hash Algorithmus ASN.1 OID> <Chunk Hash>

Reply: <reply code 3 byte>

Kode	*Mnemonik*	*Bedeutung*
201	TROK	Der Hinweis auf den Datei-Index wurde akzeptiert.
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
501	NETFAIL	Ein, nicht näher spezifizierter Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter lokaler Fehler ist aufgetreten
510	NOIMPL	Dieser Befehl ist nicht implementiert, die Information wird ignoriert

B.3.5 Chunk Location (CHUNKLOC)

Mit diesem Befehl meldet der Client dem Server, das er einen Chunk erhalten hat.

Die IP-Adresse ist die Adresse des Knotens der den Chunk verwahrt. Der Server kann die Nachricht ablehnen, wenn die IP-Adresse mit welcher der Client die Verbindung aufgebaut hat sich von der gemeldeten Adresse unterscheidet. Dann muß der Server aber mit dem Fehlerkode `NOTIP` (421) antworten.

Send: CHUNKLOC <Flags: 1 Byte> <IPv4-Address: 4 oder 16 Byte> <Hash OID len: 1 byte> <Hash Algorithmus ASN.1 OID> <Chunk Hash>

Reply: <reply code 3 byte>

Flags ist eine mit bitweisem-Oder Verknüpfte Menge einer oder mehrerer folgender Werte:

0x1	IPv6: Die Addresse ist 16 Byte lang und stellt eine IPv6 Adresse dar. Ist dieses Flag nicht gesetzt ist die Adresse 4 Byte lang und stellt eine IPv4 Adresse dar
0x2	(nicht benutzt)
0x4	(nicht benutzt)
0x8	(nicht benutzt)
0x10	(nicht benutzt)
0x20	(nicht benutzt)
0x40	(nicht benutzt)
0x80	(nicht benutzt)

Folgende Fehlerkodes können auftreten:

Kode	*Mnemonik*	*Bedeutung*
201	TROK	Der Server hat die Nachricht entgegen genommen..
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
421	NOTIP	Die IP Adresse des Senders stimmt nicht mit der angegebenen Adresse überein.
501	NETFAIL	Ein, nicht näher spezifizierter, Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter, lokaler Fehler ist aufgetreten
510	NOIMPL	Dieser Befehl ist nicht implementiert, die Information wird ignoriert

B.3.6 Set Watcher (SWATCH)

Dieser Befehl ist Bestandteil des Chunk Notifikatio Broadcast (CNB) Protokolls. Wotan verwaltet, verteilt auf einem Subset aller Knoten eine Tabelle in der Chunks-IDs (als Schlüssel) und Listen von IP-Adressen (als Wert) von Watchern. Watcher sind Knoten die an Informationen über diese Chunks interessiert sind. Bei Benachrichtigungen mit dem CHUNKLOC (Abschnitt B.3.5) Kommando werden diese an alle Knoten geschickt die in der Liste stehen.

Der Server kann die Annahme des Befehls verweigern wenn die IP-Adresse des Senders nicht mit der im Befehl eingetragenen Adresse übereinstimmt. Dann muß der Server aber mit dem Fehlerkode `NOTIP` (421) antworten.

Send: SWATCH <Flags: 1 Byte> <IPv4-Address: 4 oder 16 Byte> <Hash OID len: 1 byte> <Hash Algorithmus ASN.1 OID> <Chunk Hash>

Reply: <reply code 3 byte>

Flags ist eine mit bitweisem-Oder Verknüpfte Menge einer oder mehrerer folgender Werte:

0x1	IPv6: Die Addresse ist 16 Byte lang und stellt eine IPv6 Adresse dar. Ist dieses Flag nicht gesetzt ist die Adresse 4 Byte lang und stellt eine IPv4 Adresse dar
0x2	(nicht benutzt)
0x4	(nicht benutzt)
0x8	(nicht benutzt)
0x10	(nicht benutzt)
0x20	(nicht benutzt)
0x40	(nicht benutzt)
0x80	(nicht benutzt)

Folgende Fehlerkodes können auftreten:

Kode	*Mnemonik*	*Bedeutung*
201	TROK	Der Watcher wurde in die Tabelle eingetragen.
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
421	NOTIP	Die IP Adresse des Sender stimmt nicht mit der angegebenen Adresse überein.
501	NETFAIL	Ein, nicht näher spezifizierter, Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter, lokaler Fehler ist aufgetreten
510	NOIMPL	Dieser Befehl ist nicht implementiert, die Information wird ignoriert

B.3.7 Remove Watcher (RWATCH)

Watcher können ihr Interesse an den Chunk-Informationen auch wieder aufgeben. Mit diesem Befehl werden sie aus der Liste gestrichen.

Der Server kann die Annahme des Befehls verweigern wenn die IP-Adresse des Senders nicht mit der im Befehl eingetragenen Adresse übereinstimmt. Dann muß der Server aber mit dem Fehlerkode `NOTIP` (421) antworten.

Send: RWATCH <Flags: 1 Byte> <IPv4-Address: 4 Byte> <Hash OID len: 1 byte>

<Hash Algorithmus ASN.1 OID> <Chunk Hash>

Reply: <reply code 3 byte>

Flags ist eine mit bitweisem-Oder Verknüpfte Menge einer oder mehrerer folgender Werte:

0x1	IPv6: Die Addresse ist 16 Byte lang und stellt eine IPv6 Adresse dar. Ist dieses Flag nicht gesetzt ist die Adresse 4 Byte lang und stellt eine IPv4 Adresse dar
0x2	(nicht benutzt)
0x4	(nicht benutzt)
0x8	(nicht benutzt)
0x10	(nicht benutzt)
0x20	(nicht benutzt)
0x40	(nicht benutzt)
0x80	(nicht benutzt)

Folgende Fehlerkodes können auftreten:

Kode	*Mnemonik*	*Bedeutung*
201	TROK	Der Watcher wurde aus der Tabelle entfernt.
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
421	NOTIP	Die IP Adresse des Sender stimmt nicht mit der angegebenen Adresse überein.
501	NETFAIL	Ein, nicht näher spezifizierter, Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter, lokaler Fehler ist aufgetreten
510	NOIMPL	Dieser Befehl ist nicht implementiert, die Information wird ignoriert

B.3.8 Get List of Watchers (GETWATCH)

Send: GETWATCH <Flags: 1 Byte> <Hash OID len: 1 byte> <Hash Algorithmus ASN.1 OID> <Chunk Hash>

Reply: <reply code 3 byte>

Wenn der Rückgabewert `TROK` (201) ist, so folgt direkt die Liste der interessierten Knoten, getrennt nach IPv4 und IPv6 Adressen:

Reply: <Anzahl der IPv4 Adresse: 3 Byte> [<IPv4 Adresse: 4 Byte>] ...<Anzahl der IPv4 Adresse: 3 Byte> [<IPv6 Adresse: 16 Byte>] ...

Kode	*Mnemonik*	*Bedeutung*
201	TROK	Der Watcher wurde aus der Tabelle entfernt.
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
501	NETFAIL	Ein, nicht näher spezifizierter, Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter, lokaler Fehler ist aufgetreten
510	NOIMPL	Dieser Befehl ist nicht implementiert, die Information wird ignoriert

B.3.9 Get Identy Node (GETIDNODE)

Mit diesem Kommando kann der Client feststellen wo sich ein Identity-Knoten befindet. Die Addresse des Server erhält der Client über das *IDDiscover* Chord Overlay-Netzwerk.

Send: GETIDNODE <Flags: 1 Byte> <Hash OID len: 1 byte> <Hash Algorithmus ASN.1 OID> <Identity Hash>

Reply: <reply code 3 byte>

Wenn der Rückgabewert `TROK` (201) ist, so folgt die Adresse eines Knotens auf der die gegebene Identity installiert ist.

Reply: <IPv4 Addresse des Identity Knotens: 4 Byte>

Kode	*Mnemonik*	*Bedeutung*
201	TROK	Der Watcher wurde aus der Tabelle entfernt.
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
501	NETFAIL	Ein, nicht näher spezifizierter, Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter, lokaler Fehler ist aufgetreten
510	NOIMPL	Dieser Befehl ist nicht implementiert, die Information wird ignoriert

Dieses Kommando kann in der momentanen Protokolvariante nicht mit IPv6 Adressen umgehen.

B.3.10 Identity Update Query (IUQUERY)

Mit diesem Kommando kann ein Identity-Knoten die eigenen Listen der Speicherobjekte mit anderen ID-Knoten dieser Identität abgleichen. Mit Hilfe des Hashes der Identität kann eine einzelne Identität ausgewählt werden, wenn auf einem Knoten mehr als eine Identität installiert ist. Das Verschlüsselungsverfahren wird vom Client bestimmt um sicher zu gehen daß das Verfahren auf beiden Knoten implementiert ist.

Send: IUQUERY <Hash OID len: 1 byte> <Hash Algorithmus ASN.1 OID> <Identity Hash> <Länge der OID des Verschlüsselungs-Alg.: 1 Byte> <OID des Verschlüsselungs-Alg.>

Reply: <reply code 3 byte>

Alle Daten der Antwort, die nach dem Kommandowort und der OID des Verschlüsselungsverfahrens übertragen werden, sind mit dem Symetric Master Key und dem angegebenen Verfahren verschlüsselt. Wenn der Reply-Code 201 ist folgt die Antwort auf die Anfrage:

Reply: <Länge der OID des Verschlüsselungs-Alg.: 1 Byte> <OID des Verschlüsselungs-Alg.> $enc_S MK$(<Länge der OID des Verschlüsselungs-Alg.: 1 Byte> <OID des Verschlüsselungs-Alg.> <Länge der OID des Hash-Alg.: 1 Byte> <OID des Hash-Alg.> <Anz der Files: 4 Bytes> <Hash des 1. Files: var. > <Schlüssel für 1. File: var. > ...)

Mögliche Reply-Codes:

Kode	*Mnemonik*	*Bedeutung*
201	TROK	Die Identität ist auf der Maschine instaliert und bereit Daten zu den Verwalteten Files zu übertragen.
401	NOHASH	Der Hash-Algorithmus ist nicht bekannt oder die Berechnung des Hashes ist aus anderen Gründen nicht möglich
404	NOCRYPT	Der Verschlüsselungs-Algorithmus ist nicht bekannt oder die Verschlüsselung ist aus anderen Gründen nicht möglich
501	NETFAIL	Ein, nicht näher spezifizierter, Fehler in der Übertragung ist aufgetreten.
503	LFAIL	Ein, nicht näher spezifizierter, lokaler Fehler ist aufgetreten
510	NOIMPL	Dieser Befehl ist nicht implementiert, die Information wird ignoriert

B.4 Dateistruktur

Chunks welche die Struktur einer Datei enthalten sind als Folge von Bytes wie folgt aufgebaut:
<Chunk-Größe, 4 Byte, little-endian>
<Community-Id, 8 Byte, little-endian>
<Länge der Datei, 8 Byte, little-endian>
<Anzahl der Bytes in des Verschlüsselungs-Oid, 1 Byte>
<Verschlüsselungs-Oid, variable Länge>
<Anzahl der Bytes in der Hash-Oid, 1 Byte>
<Hash-Oid, variable Länge>
<Anzahl der Chunks in der Datei, 4 Byte, little-endian>

Danach folgen die Hashes der Chunks in der Datei. Ihre Länge wird durch den verwendeten Hash-Algorithmus bestimmt.

Tabelle B.1 gibt Informationen für mögliche Verschlüsselungsalgorithmen an. Eine OID Länge von 0 signalisiert einen unverschlüsselten Chunk. Für jeden Chunk wird die Verschlüsselung separat initialisiert.

Alg/Mode	*OID (ASN.1/DER)*	*Key-length*	*Java Cipher Identifier*
AES-CBC (128 Bit Key, default)	2 16 840 1 101 3 4 1 2 (06:09:60:86:48:01:65:03:04:01:02)	128 Bit	`AES/CBC/PKCS5Padding`
AES-CBC (192 Bit Key)	2 16 840 1 101 3 4 1 22 (06:09:60:86:48:01:65:03:04:01:16)	192 Bit	`AES/CBC/PKCS5Padding`
AES-CBC (256 Bit Key)	2 16 840 1 101 3 4 1 42 (06:09:60:86:48:01:65:03:04:01:2A)	256 Bit	`AES/CBC/PKCS5Padding`

Tabelle B.1: OIDs für Verschlüsselungsalgorithmen

B.5 Benutzer Identität

Eine Benutzer Identität besteht aus einem RSA Schlüsselpaar und einem gewählten Benutzernamen des Benutzers. Dieser Benutzername ist ein UTF-8 kodierter String von beliebiger Länge.

Der User Identifier berechnet sich aus dem SHA-384 Hash des Schlüsselpaares in PKCS-8 Format und dem Benutzernamens.

B.6 Notizen und Anmerkungen

In der vorliegenden Version enthält das Protokol noch einige Lücke und Schwächen. Ein paar Überlegungen für Weiterentwicklungen sind im folgenden notiert. Die Reihenfolge ist zufällig und stellt keine Gewichtung dar.

In der momentanen Version des Protokolles ist eine Verschlüsselung noch nicht spezifiziert[1], aber das eine der intendierten Klasse von Zielsystemen auch mobile Geräte mit begrenzter Rechenleistung und Energievorrat sind, sollte asymetrische Crypto auf dem Client wenn möglich vermeiden werden. Es ist die Möglichkeit für den Einsatz von HMAC's zu untersuchen.

Im Moment sind noch keine Timeouts definiert[2]

Eine Übetragung von Gewichtungen von Dateien ist nocht nicht unterstützt. Da diese Informationen aber von der jeweiligen Policy abhängen, wird das Protokol wahrscheinlich noch um einen Policy spezifischen Datenblock erweitert werden müssen. Da potentiell jede Policy eigene Daten definieren muß, sollte der Mechanismuß entsprechend flexibel sein.

[1] "Sicherheit kommt später"

[2] Macht aber nix, die Referenzimplementation unterstützt auch noch keine Timeouts

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§25(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 11. Oktober 2006 Lutz Behnke