ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Reto Keiser

Distributed Content-based Accounting and Charging in P2P

Diploma Thesis DA-2003.33
Summer Term 2003

Tutor: David Hausheer

Supervisor:
Prof. Dr. Burkhard Stiller

7.9.2003

TIK Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Abstract

The last few years, Peer-to-Peer Networks (P2P) got very popular. Mainly the possibility to share multimedia content helped these networks to spread within the Internet.

Even if the reliability of the connected peers and the quality of the data is not as good as in classical client-server systems, there are still many advantages. The content is greatly replicated and therefore it is not necessary, that all the customers need to share the upload bandwidth of one server. The traffic is spread within the network in a more equal way resulting in an increase of throughput and improvement of response time. If one source of a file is temporarily unavailable, it is possible to get it from a different peer.

In reality this scenario gets a bit worse: many participants of the network are egoistic and load content from the network without contributing anything on their own. On the other hand, there are altruistic peers who share a lot of content. That makes the system having similar characteristics like a client-server system, what actually should be avoided.

The goal of this project is to design a system which motivates a many participants to share their own content in order to balance the network traffic between the peers. To achieve this, a virtual currency is introduced, whose units are called "tokens". Every user who wants to get content from the network needs these tokens as kind of money. Therefore, the motivation to earn some tokens is big, what the peers can achieve by uploading some content. It is even possible that a virtual market with many competitors is emerged.

Besides that, a method is introduced, which makes possible, that a customer peer can send download offers to the provider peers. This so-called auction-based method has the advantage, that the provider peer can just accept and acknowledge these offers but upload these files later, when its upload bandwidth is not fully used. This method works in tandem to the classical downloads with predefined prices.

The customer peers have to adapt to the market, to keep the system running efficiently. If the prices are outside some certain bounds, the number of earned tokens will decrease and hence the profit might suffer. The client needs to be implemented in a way, so that it is still working properly in an environment which is changing over time.

Inhalt

1. Einleitung

Peer-to-Peer Netzwerke (P2P) haben in den letzen Jahren stark an Popularität gewonnen. Vor allem die Möglichkeit, Multimediadateien untereinander auszutauschen, trug viel zur Verbreitung dieser Dienste bei.

Auch wenn die Zuverlässigkeit der einzelnen Knoten und die Qualität der Daten durch die Verteiltheit nicht so gross ist wie bei klassischen Client-Server Systemen, bietet die Peer-to-Peer Technologie trotzdem einige Vorteile. Da die Dateien in mehrfacher Ausführung im ganzen Verbund verteilt gespeichert sind, verbessert sich die Performance und Stabilität des Systems. So müssen sich beispielsweise nicht alle Interessenten einer Datei die Uploadbandbreite eines Servers teilen, sondern können diese bei vielen verschiedenen Peers beziehen, welche sonst ihre Uploadbandbreite nicht ausschöpfen würden. Dadurch verteilt sich der Traffic besser im Netzwerk und die Antwortzeiten verkürzen sich. Falls eine mögliche Quelle einer Datei ausfällt, kann einfach auf eine andere ausgewichen werden.

Dies sind die idealen Eigenschaften eines Peer-to-Peer Netzwerks. In der Realität sieht es oft etwas anders aus: Sehr viele Teilnehmer eines solchen File-sharing Netzwerkes denken nur an sich selber und beziehen Content aus dem Netzwerk, ohne selber etwas beizutragen. Andere altruistische Peers steuern der Community viel Content bei, ohne selber etwas aus dem Peer-to-Peer Netzwerk zu laden. Dadurch erhält das System wieder eine ähnliche Charakteristik wie ein klassisches Client-Server System, was eigentlich vermieden werden sollte.

Ziel dieser Arbeit ist es nun, einen Mechanismus zu entwerfen, welcher möglichst viele Benutzer des Systems dazu motiviert, selber sich aktiv als Provider Peers zu beteiligen, damit der Netzwerkverkehr ausbalanciert wird und der mögliche Durchsatz ansteigt. Um dies zu bewerkstelligen, wird eine virtuelle Währung geschaffen, deren Einheit oder Geldstücke in dieser Arbeit "Token" genannt werden. Diese Token sollten die Benutzer dazu motivieren, selber Content anzubieten, damit sie mit den erhaltenen Tokens selber wieder Dateien kaufen können. Da jeder Peer, welcher Files aus dem Peer-to-Peer Netzwerk beziehen will, für diesen Zweck Tokens benötigt, ist die Motivation dementsprechend gross, selber Dateien zur Verfügung zu stellen. Dadurch ist es durchaus möglich, dass sich ein virtueller Markt bildet und untereinander Konkurrenz entsteht.

Weiter wird eine Methode untersucht, bei welcher nicht nur der Provider Peer angibt, zu welchem Preis eine bestimmte Datei zu haben ist, sondern dass auch der Customer Peer die Möglichkeit hat, eine Downloadofferte an den Provider Peer zu richten. Diese so genannte Auction-based Methode hat zudem den Vorteil, dass der Provider Peer diese Offerten vormerken kann, um dann diese Dateien später zum vereinbarten Preis abzugeben. Dies wird er dann tun, falls einmal seine Uploadbandbreite nicht vollständig ausgelastet ist und sonst keinen Profit abwerfen würde. Diese Methode läuft parallel zum klassischen Verkauf von Content und erfordert vom Benutzer ein taktisches Abwägen der Vor- und Nachteile.

Damit dies alles reibungslos funktioniert, muss der Client sich jeweils an den Markt anpassen, damit seine Preise im Rahmen liegen, um die eigene Uploadbandbreite so stark auszulasten, dass der Profit optimal ist. Da sich die Strategien der Peers ändern können und das Verhalten der Benutzer im Internet sich mit der Zeit wandelt, sollte der Client so ausgelegt sein, damit diese Aufgabe auch in Zukunft noch zuverlässig erfüllt wird.

2. Grundlagen

In Peer-to-Peer Netzwerken kommunizieren die einzelnen Peers direkt miteinander. Je nach dem wie stark dieser Netzwerkverkehr verteilt ist, desto besser ist der Durchsatz des ganzen Verbundes. Das Ziel der vorliegenden Arbeit ist nun, dieser Netzwerkverkehr so zu verteilen, dass die einzelnen Peers nicht zu stark belastet werden, aber trotzdem genügend viel Content über das Netzwerk geschickt werden kann.

2.1. Ausbalancierung der Last

Wenn Server Dateien zur Verfügung stellen, welche von vielen Benutzern heruntergeladen werden, müssen diese Server für jeden einzelnen Client die ganze Datei erneut ins Internet schicken. Das Bottleneck bildet in diesem Falle die direkte Anbindung dieses Servers an das Internet, welche in einem bestimmten Zeitraum nur eine begrenzte Menge an Daten transportieren kann. Diese Verbindung zum Internet ist für den Server mit Kosten verbunden, welche oft mit wachsender Datenmenge ansteigt.

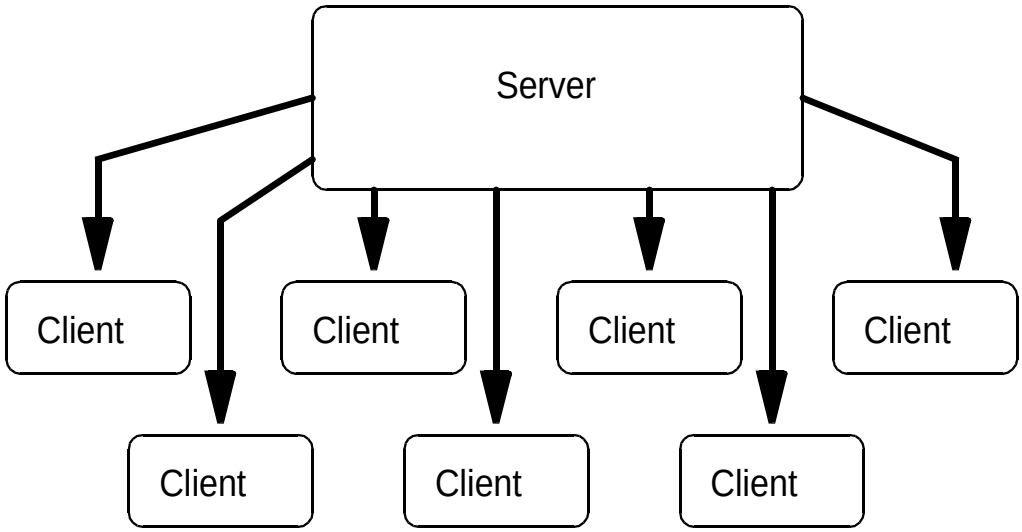

Abb.1: Grosse Belastung eines Peers, falls viele andere Peers darauf zugreifen.

Falls die Daten nur bei diesem einen Server erhältlich sind, kann man nicht garantieren, dass sie stets verfügbar sind. Ein einzelner Server ist anfällig gegen Einflüsse von aussen wie zum Beispiel Netzwerkunterbrüche, Systemabstürze und Denial of Service Attacken. Auf der anderen Seite kann man davon ausgehen, dass die Integrität dieser Daten in Ordnung ist, sofern die Reputation dieses Servers gut ist.

Die Idee eines Peer-to-Peer Netzwerks ist die räumliche Verteilung und Replikation der Daten innerhalb des Netzwerks. Jeder Peer kann sowohl als Anbieter als auch Empfänger von Daten auftreten. Dabei ist die gesendete und empfangene Menge an Daten innerhalb des Netzwerks gleich gross. Je mehr Leute selber Daten anbieten, desto besser verteilt sich der Traffic im ganzen Netz. Dazu kommt noch, dass in diesem Falle die Kopie einer bestimmten Datei gleichzeitig bei vielen verschiedenen Peers zu finden ist. Dies steigert die Wahrscheinlichkeit, dass diese Datei von guter Qualität ist, weil sonst ein grosser Teil der Benutzer den Fake erkannt und gelöscht hätten anstatt, ihn weiter anzubieten.

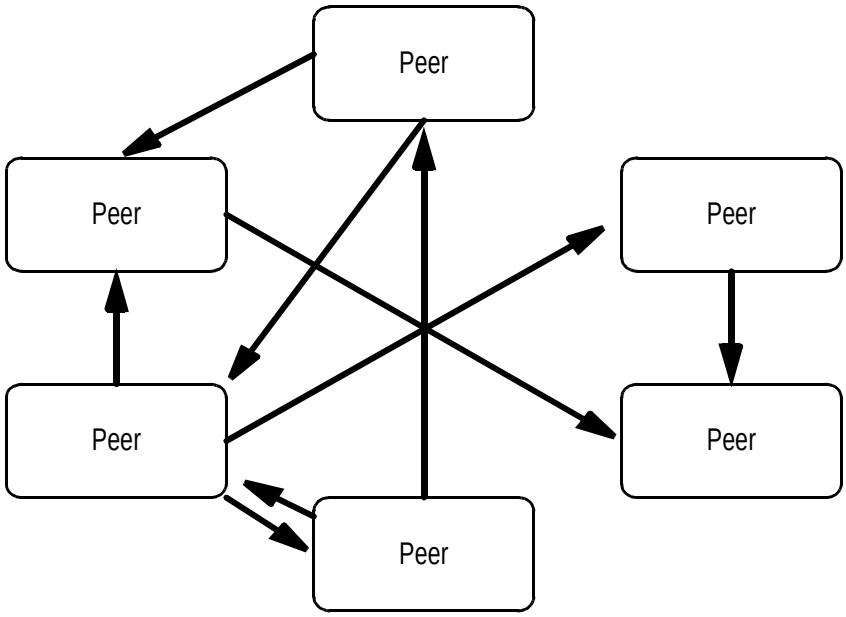

Abb 2: Wenn die Traffic verteilt wird, sinkt die Belastung eines einzelnen Peers

Im Idealfall sendet ein Peer in etwa dieselbe Menge Daten, wie er von den anderen Peers empfängt. Dadurch wird die Upload- und Downloadbandbreite der einzelnen Peers so gut wie möglich ausgelastet. Falls die ins Netz gesendete Menge Daten nicht die empfangene Menge signifikant übersteigt, bleiben im Vergleich zu Serversystemen auch die Kosten im Rahmen. Auch in Peer-to-Peer Netzwerken wird es uneigennützige Teilnehmer geben, die sehr viel Content zur Verfügung stellen und mit hoher Bandbreite hinaufladen, ohne dass sie selber viele Dateien von anderen Teilnehmern beziehen. Diese Server nehmen freiwillig die möglichen Mehrkosten in Kauf. Solche Peers nützen natürlich auch dem Durchsatz des ganzen Systems, aber es wäre nicht gut, wenn diese die einzigen Peers wären, welche Content zur Verfügung stellen.

Es ist schwierig, möglichst viele der Peers dazu zu bringen, selber Content zur Verfügung zu stellen, wenn für sie nicht direkt ein Vorteil herausschaut [3]. Aus diesem Grund wird versucht, eine Motivation zu schaffen, selber aktiv am Dateiaustausch teilzunehmen [8].

2.2. Motivation zum Anbieten von Content

Grundsätzlich gibt es drei verschiedene Arten von Motivation, sich selber aktiv im Netzwerk zu beteiligen:

- Belohnen von Benutzern, die Uploadbandbreite zur Verfügung stellen
- Bestrafen von Free-loadern, welche selber nichts beitragen, indem sie gewisse Nachteile gegenüber anderen Benutzern haben.
- Der Client so implementieren, damit die Benutzer verpflichtet sind, auch Content ins Netz zu laden.

Ob die letzten zwei Punkte wirklich als echte Motivation gelten, kann man bezweifeln. Eigentlich sind dies eher Strafen, welche egoistisches Verhalten unattraktiv machen. Der Unter-

schied zwischen Belohnen und Bestrafen ist fliessend. So kann man es so darlegen, dass zum Beispiel Leute, die Content bereitstellen, im Durchschnitt einen höheren Durchsatz beim Download haben, andererseits kann dies auch so ausgelegt werden, dass die Downloadrate derjenigen Benutzer ohne Upload begrenzt wird. Dennoch ist es da Ziel, das schlussendlich möglichst viele Benutzer aktiv am System teilnehmen [7].

In dieser Arbeit wird versucht, das Problem mit einer virtuellen Währung zu lösen. Diese kann sowohl als Motivation betrachtet werden, als auch als Mittel, damit jene Leute, die nichts zum System beitragen, auch nicht profitieren können [12]. Dabei kommt es nicht darauf an, womit ein Teilnehmer diese Tokens verdient hat. Stammen sie von einem Upload, hat er bereits einen Beitrag zum Netzwerk geleistet; sind sie aber irgendwoher zugekauft, hat dieser mit dem Geld jemandem geholfen, die Unkosten zu decken. Die Tokens sind also ein direkter Gegenwert, so dass die Nutzenbilanz bei den Downloads einigermassen ausgeglichen ist. Wenn die Tokens im Netzwerk wieder ausgegeben werden, bleiben sie im Umlauf und geben einem anderen Nutzer die Möglichkeit, Content aus dem Netz zu beziehen.

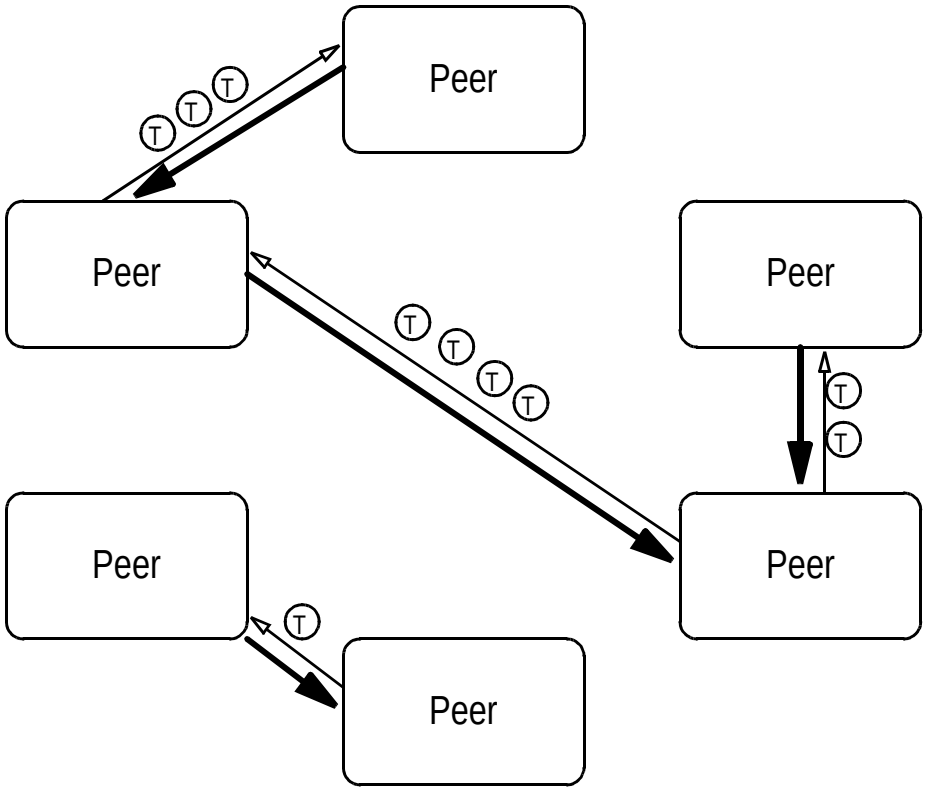

Abb 3: Für den Gegenwert jeder heruntergeladenen Datei bezahlt der Customer Peer eine entsprechende Anzahl Tokens.

Da Benutzer des Internets momentan tendenziell nicht bereit sind, echtes Geld im Netzwerk auszugeben, wird das auch bei den Peer-to-Peer Netzen nicht der Fall sein, und die meisten Teilnehmer werden wahrscheinlich die Tokens mit Hilfe von Uploads verdienen wollen. Dies erhöht die Anzahl Quellen für einen Download und den Datendurchsatz im Peer-to-Peer Netzwerk allgemein. Dies ist genau der Zweck, welcher die virtuelle Währung im Netzwerk erfüllen soll.

2.3. Optimierung des Gewinns und der Auslastung der Internetanbindung

Die meisten Benutzer eines Peer-to-Peer Netzwerkes werden dieses aufsuchen, da sie auf der Suche nach Content sind. Der Hauptzweck ist also die Beschaffung der Daten. Wenn in diesem Falle die Ausbalancierung der Last mittels einer virtuellen Währung gelöst wird, so benötigen diese Benutzer eine gewisse Anzahl Tokens, um die gewünschten Dateien aus dem Netz zu laden. Falls sie den Client noch öfters am Laufen haben, mit dem Zweck, dass die anderen Benutzer des Netzwerks auf die eigenen freigegebenen Dateien zugreifen können, ohne dass sie selber Content aus dem Netzwerk beziehen, besteht kaum eine Gefahr, dass es an Tokens mangelt.

Für diese Benutzergruppe, die jedoch nur das Minimum das Content ins Netzwerk abgibt, um die eigenen Downloads finanzieren zu können, ist es wichtig, taktisch im Markt aufzutreten. So kann jeder Benutzer selber aussuchen, wo er die Daten herholt und wie viel er dafür bezahlen will. Wenn man eine günstige Quelle aussucht, muss man aber stets damit rechnen, dass dort die Nachfrage dementsprechend gross ist und man länger auf einen Download warten muss. Das Ziel ist es nun, einen guten Mittelwert zwischen Preis und Wartezeit zu finden. Auf der anderen Seite hat ein Benutzer, der Content anbietet, die Möglichkeit, den Preis der Dateien zu bestimmen. Falls dieser zu hoch ist, finden sich keine Interessenten für die entsprechenden Dateien; ist der Preis jedoch zu tief, ist die Nachfrage zwar hoch, aber der Benutzer verdient weniger pro Download.

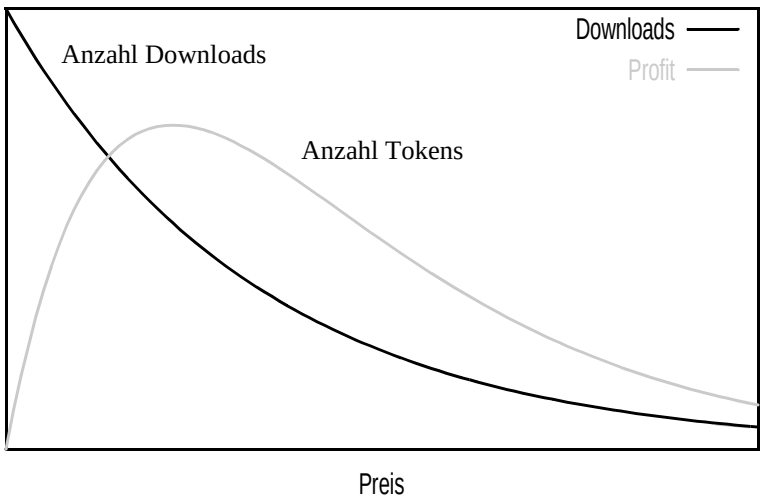

Abb 4: Je billiger die Datei, desto mehr Downloads können damit erzielt werden. Da die Bandbreite aber beschränkt ist, stellt sich das Optimum in der mittleren Preislage ein.

Nun kann dem Benutzer natürlich nicht zugemutet werden, dass er im Netzwerk die Preise aller Anbieter vergleicht, wenn er ein File zur Verfügung stellt, das im Netzwerk bereits von anderen Leuten angeboten wird. Diese Preisvergleiche und die Festlegung der Preise für den eigenen Content ist nun die Aufgabe des Clients. Dadurch muss sich der Benutzer nicht darum kümmern, während der Client die Preise permanent dem Markt anpassen kann.

3. Konzepte

Im folgenden Kapitel werden die verschiedenen Ideen hinter dem Projekt der Lastbalancierung erläutert. Dabei kommt es nicht nur auf die Ansätze an, die Teilnehmer eines Peer-to-Peer Netzwerks zu motivieren, aktiv daran Teilzunehmen, sondern auch das optimale Ausnutzen der Bandbreite um den Durchsatz zu maximieren.

3.1. Virtuelle Währung für den Dateiaustausch

Um die Benutzer zu motivieren, selber Content ins Peer-to-Peer Netzwerk abzugeben, kann der Gebrauch von einer Belohnung gemacht werden, die sie für jeden erfolgreichen Upload erhalten. Diese kann die Form von Credits, einem besseren participation Level [9] oder von virtuellem Geld [12] haben. Der Zweck ist stets derselbe. In der vorliegenden Arbeit wird dies mit einer virtuellen Währung, den so genannten Tokens erreicht.

3.1.1. Tokens als Währungseinheit

Um den Dateiaustausch innerhalb des Netzwerks möglichst fair zu gestalten, wurde in diesem Falle eine lokale Währung eingeführt. Die Grundeinheit davon nennen wir der Einfachheit halber mal ein "Token", also ein Zeichen oder Element, das für den Zahlungsverkehr gebraucht wird. Der Einfachheit halber nehme ich für die Mehrzahl einfach die englische Pluralform "Tokens". Ein solches Token besteht grundsätzlich aus einem Datenpaket, welches Informationen enthält. Wer im Besitz dieser Information ist, besitzt so quasi dieses Geldstück, und kann mit diesem weiter einkaufen. In der Regel wird das aber nicht gemacht, denn die Person, von welcher man das Token erhalten hat, kennt dieses auch, und könnte es sofort an einer anderen Stelle nochmals ausgeben, ohne dass die erste Person davon etwas merkt. Aus diesem Grund hat es Sinn, ein erhaltenes Token so früh wie möglich bei einer zentralen Stelle einzutauschen. Auf diese Weise kann man zum einen sicherstellen, dass das Token von der Person, von welcher man das Token erhalten hat, nicht schon einmal ausgegeben wurde. Zum anderen erhält man beim Eintausch dieses Tokens ein anderes Token, was sicherstellt, dass erstere Person in Zukunft nicht mehr in der Lage sein wird, dieses Token nochmals als Zahlungsmittel zu verwenden.

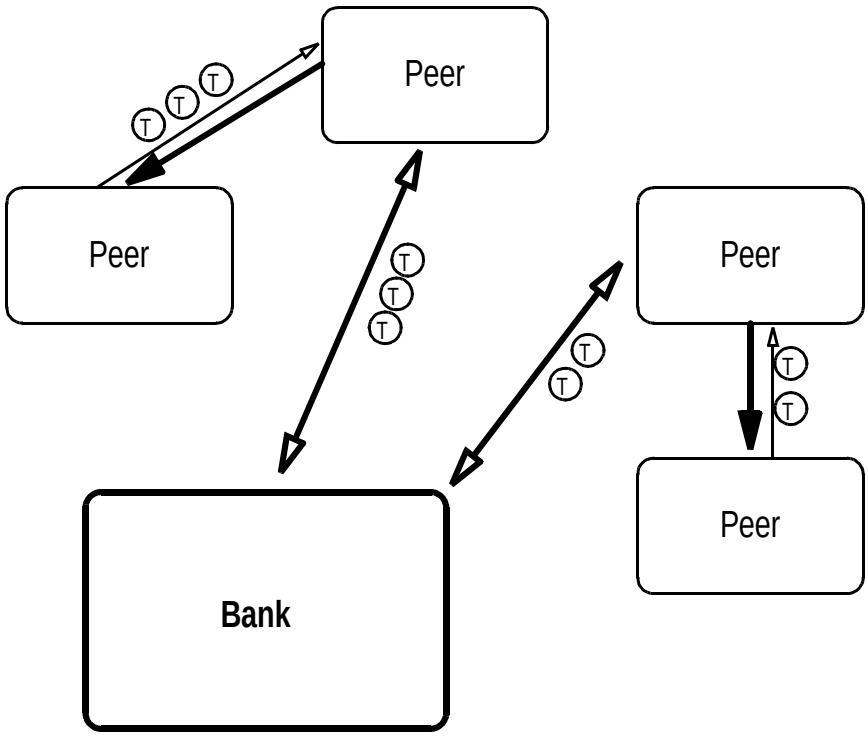

Abb 5: Die Bank übernimmt das Recycling der Tokens um dem Double spending vorzubeugen

Tokenausgabestelle als Bank im Netzwerk

Damit es im Netzwerk zu keinen Problemen kommt, sollte die zentrale Tokenausgabestelle von einer vertrauenswürdigen Instanz geführt werden. Diese hat dann die Möglichkeit beliebig Tokens ins Netzwerk auszugeben und wieder zurückzunehmen. In der Realität sollte aber diese Bank darauf achten, dass die Zahl der Tokens (*Tokens*) in Relation zur Anzahl Benutzer (*Users*) des Netzwerks in etwa konstant bleibt. Es ist aber auch möglich, die Zahl der insgesamt angebotenen Dateien (*Files*) und die durchschnittliche Bandbreite (*bwith*) in diese Rechnung hineinfliessen zu lassen, denn je grösser diese Zahl ist, desto mehr Dateien werden gleichzeitig getauscht. Man könnte die Anzahl Tokens im Filesharingnetzwerk mit Hilfe folgender Formel approximieren:

$$Tokens \cong const \cdot Users \cdot \sqrt{bwith \cdot Files}$$

Zudem wird noch *const* als neutrale Konstante eingeführt, welche abhängig vom Netzwerk und der Tokenausgabestelle ist. Wenn aber zu viele Peers im Netzwerk die Tokens bloss sammeln und nicht mehr ausgeben, könnte Trotzdem eine Knappheit entstehen, welche die Tokenausgabestelle beheben sollte.

Der Nachteil einer zentralen Ausgabe ist, dass das System an dieser Stelle angreifbar wird. Mögliche Attacken im Internet könnten sich gegen diese Stelle richten und so das ganze Filesharing Netzwerk lahm legen. Auch ein Systemabsturz oder andere Ereignisse könnten so die Integrität des ganzen Verbundes gefährden. Wenn anstelle eines zentralen Servers mehrere benutzt werden, erhöht sich die Stabilität des ganzen Netzwerks; in diesem Falle müssen sämtliche Instanzen der Tokenausgabestellen vertrauenswürdig sein. Die Zahl der möglichen Schwachstellen nimmt zu, weil bereits ein schwaches Glied in der Kette die Sicherheit des

ganzen gefährden kann. Zudem müssen sich diese Server stets synchronisieren, damit beispielsweise nicht das gleiche Token an verschiedenen Instanzen der Tokenausgabestellen eingelöst werden kann.

3.1.2. Eigenschaften der Tokens

Die Implementation der Tokens wird bewusst noch offen gelassen. Im Prinzip ist es möglich, die Tokens beim Austausch zu verschlüsseln und zu signieren, damit sie nicht beliebig weitergegeben werden können, sondern nur der rechtmässige Empfänger diese einlösen kann. Je nachdem, ob noch Informationen über die Identität des Senders oder Empfängers im Token gespeichert werden, könnte das datenschutztechnisch ein Problem sein.

Ob dieser Mehraufwand an Rechenkapazität Sinn macht, ist offen, denn in der Regel hat ein einzelnes Token nicht so viel Wert. So könnte es durchaus genügen, dass ein Token eine Art von One time Pad darstellt, also eine mehr oder weniger zufällige Folge von Bits, welche von einer aussenstehenden Person nicht erraten werden kann. Durch das Kennen dieser zufälligen Folge kommt man in den Besitz des Token. In diesem Falle haben die Tokens eine ähnliche Eigenschaft wie Bargeld: Es ist anonym, und wenn es jemand irgendwo findet, so kann er es beliebig benutzen. Dadurch wird es aber nötig, diese Tokens zu verschlüsseln, wenn sie auf dem lokalen Computer abgespeichert werden, damit niemand Zugriff darauf hat. Nichtsdestotrotz weiss die Ausgabestelle, an wen sie welche Tokens abgegeben hat.

Weiter ist es möglich, die Tokens mit einem Ablaufdatum zu versehen, damit sie an diese Person zurückgehen, welche sie von der Ausgabestelle bezogen hat. Dies hat der Vorteil, dass zufällig gelöschte Tokens nicht für immer verschwunden sind.

Anstelle des Bentuzens von Tokens, kann die Bezahlung innerhalb des Peer-to-Peer Netzwerks mit Hilfe einer virtuellen Bank gelöst werden [9]. Zum Bezahlen eines Downloads veranlasst der Customer Peer eine Überweisung der entsprechenden Anzahl Tokens direkt an den Provider Peer. Dabei bleibt das "Geld" stets innerhalb der Virtuellen Bank, nur der Besitzer wechselt.

3.2. Verteilung der Tokens im Netzwerk

Da im Netzwerk Dateien ausgetauscht werden, hat dies zur Folge, dass auch die Tokens den Besitzer wechseln. Wenn ein Peer mit einer grossen Bandbreite Daten anbietet, aber nur selten eine Datei herunterlädt, wird dieser Peer mit der Zeit Tokens sammeln, welche nur im viel kleineren Masse wieder ins Netzwerk zurückfliessen. Diese Tokens fehlen dann im ganzen Verbund, was im schlimmsten Fall damit enden kann, dass sich alle Tokens bei einigen wenigen Peers ansammeln. Auf der anderen Seite werden die Peers mit geringer Bandbreite für jeden Download die entsprechende Anzahl Tokens zusammensammeln müssen.

3.2.1. Aufteilung neuer Tokens

Eine Variante, die Anzahl Tokens, welche sich gerade im Umlauf befinden, konstant zu halten, ist das Verteilen von neuen Tokens. Da jedes Token aus Sicherheitsgründen nur einmal benutzt werden kann, und dann von der Bank eingetauscht werden muss, hat diese relativ genau den Überblick, wie viele Tokens sich effektiv noch im Umlauf befinden, und wie viele von bestimmten Peers gesammelt werden. Sobald die Anzahl im Umlauf befindenden Tokens im Verhältnis

zur Anzahl Benutzer unter einen bestimmten Wert sinkt, hat die Bank die Möglichkeit, neue Tokens in Umlauf zu bringen. Die Frage ist nur: wie? Wer diese Tokens erhält, kriegt eigentlich zusätzlich Geld, was aber je nach Situation nicht fair gegenüber den anderen ist. Es gibt hier mehrere Möglichkeiten, wie diese Tokens verteilt werden; zum Beispiel wäre eine Gleichverteilung unter allen Benutzern möglich. Eine andere Möglichkeit wäre die Bevorzugung solcher User, die viel zum System beitragen.

Im Prinzip sollte bei der Verteilung der Tokens keine Inflation entstehen, da die Tokens nur in einem so grossen Rahmen verteilt werden, dass die Zahl der sich im Umlauf befindenden Tokens möglichst konstant bleibt.

Wenn jedoch ein Peer für längere Zeit mit vielen Uploads Tokens sammelt und diese dann plötzlich wieder in den Umlauf bringt, würde dies eine Abwertung zur Folge haben, da dieser Peer die Mittel hat, viele Tokens in einen Download zu investieren, und so die anderen Nutzer zwingen, dasselbe zu tun, damit sie in diesem neuen Gleichgewicht die Chance haben, sich einen Download zu ergattern. Wie man in diesem Falle die Tokens wieder aus dem System entfernt, ist eine andere, schwierigere Frage. Die Token haben nämlich einen Wert, und wenn sie die Bank wieder entfernen möchte, müsste sie dementsprechend investieren. Eine Möglichkeit wäre, einfach so lange mit Verteilen von neuen Tokens zu warten, bis sich das ursprüngliche Gleichgewicht wieder eingestellt hat. Dies sollte im Prinzip passieren, wenn mit der Zeit die Datenmenge anwächst und die Zahl der Benutzer nicht abnimmt.

Eine andere mögliche Lösung wäre, dass die Bank beim Verkauf von neuen Tokens das dabei verdiente Geld als Reserve behält, um im Falle einer Inflation dieses Geld für den Rückkauf von Tokens einzusetzen.

3.2.2. Tokens mit Geld kaufen und verkaufen

Im Prinzip ist es möglich, dass man stets gegen reales Geld Tokens kaufen oder verkaufen kann. In diesem Falle haben auch Leute, die kein Content bereitstellen wollen, die Möglichkeit, Tokens zu beschaffen, um damit Dateien herunterzuladen. Peers, welche Dateien zum Download bereitstellen, können die so erhaltenen Tokens gegen Geld eintauschen und so die Unterhaltungskosten zu einem Teil decken. Auf diese Weise ist es eher möglich, die Zahl der im Umlauf befindenden Token zu verändern.

Der Nachteil an dem Ganzen ist, dass das Ganze dadurch einen kommerziellen Beigeschmack bekommt. Wenn zum Beispiel Open Source Dateien gegen Tokens zum Download angeboten werden, ist nicht mehr klar, ob der Preis nur die Unkosten deckt, oder ob versucht wird, mit solchen Dateien Gewinn zu erwirtschaften.

Die meisten Menschen sind von der Kultur so geprägt, dass sie stets den eigenen Profit zu maximieren versuchen. Sobald es in einem solchen Netz etwas zu verdienen gibt, kann davon ausgegangen werden, dass dies solche Nutzer anzieht, welche dies ausnutzen wollen. Zwar wird durch die Konkurrenz der Preis auf einem tiefen Niveau gehalten, dieser wird aber trotzdem höher sein, als wenn es die Möglichkeit zum Eintauschen der Tokens gegen Geld nicht gäbe. Wenn diese Peers nun die ganze Zeit Dateien zum Download anbieten und dann die Tokens verkaufen, sinkt mit der Zeit die Zahl der Tokens im ganzen Verbund. Zudem müssen die Tokenausgabeserver für die Tokens, welche in reales Geld getauscht werden, bezahlen. Daraus würde ein variabler Wechselkurs resultieren, so dass der Wert der Tokens vom Angebot und der Nachfrage abhängt.

Je nach Höhe des Preises für die Tokens, werden die Peers mit einer langsamen Anbindung sich

überlegen, ob es wohl mehr Sinn machen würde, selber Dateien bereitzustellen oder aber schnell ein paar Tokens zuzukaufen. Sobald zum Beispiel eine Modemleitung mehr kostet, als sie Tokens bringt, lohnt sich der direkte Zukauf. In diesem Falle würde aber das Prinzip der Lastbalancierung mittels Tokens versagen, weil das System wieder auf ein Client - Server - Modell hinausläuft. Glücklicherweise kann aber davon ausgegangen werden, dass die Zahl der Leute mit einer langsamen Internetanbindung nicht so gross ist und sich tendenziell verkleinert.

Sobald diese Möglichkeit, Tokens gegen Geld einzutauschen sich in einem grösseren Stil durchsetzt, stellt sich die Frage, wie sich das mit den Steuersystemen der einzelnen Länder vereinbaren lässt. Dies ist aber eher eine politische Frage und wird in dieser Arbeit nicht weiter behandelt.

3.2.3. Tokens gegen andere Dienstleistungen eintauschen

Einer Kommerzialisierung des Netzes kann auch vorgebeugt werden, indem die Tokens zwar nicht gegen Geld, dafür aber in andere Dienstleistungen eingetauscht werden können. Eine Möglichkeit wäre zum Beispiel Rechenkapazität. In der Regel haben einzelne Anwender im Vergleich zu grösseren Servern relativ gesehen mehr ungenutzte Rechenleistung, da diese nur bei bestimmten Programmen ausgeschöpft wird. Demgegenüber sind Server aus Kostengründen meist so dimensioniert, dass sie mehrheitlich gut ausgelastet sind, also nicht mehr so viel Kapazität übrig haben. In diesem Falle können nun die Server bei den anderen Peers Rechenkapazität gegen Tokens einkaufen, da diese durch die breitere Anbindung sowieso mehr Dateien zum Download anbieten können. Auch Firmen, die Rechenkapazität für eigene Dateien benötigen, haben die Möglichkeit, sich dem Netz anzuschliessen, auf einem sowieso schon vorhandenen Webserver einige Dateien gegen Tokens zur Verfügung zu stellen, und damit bei den Nutzern Processingpower zuzukaufen.

Natürlich kann dieses Konzept nicht nur für Rechenleistung gebraucht werden, sondern vielleicht auch für Speicherkapazität (Auch wenn hier berechtigte Zweifel auftreten können, wie sicher so ein verteiltes Speichermedium gegen Ausfälle ist).

Im Prinzip ist es auch möglich, beide Verfahren zu kombinieren und sowohl Geld als auch Rechenleistung als Gegenleitung für die Tokens zu benutzen. Dadurch wird das Ganze wieder kommerziell, aber auch flexibler.

3.2.4. Gewichtung der Tokens in Abhängigkeit von der Bandbreite

Im bisher skizzierten System haben Peers mit zeitlich unbeschränkter Anbindung und grosser Bandbreite Vorteile gegenüber Benutzer mit Telefonverbindungen. Dadurch ist es für letztere viel aufwändiger, an Tokens zu kommen.

Um den kleineren Peers die Motivation, Dateien anzubieten, nicht zu nehmen, besteht die Möglichkeit, dass die Tokens abhängig von der Bandbreite gewichtet (*bias*) werden. In diesem Falle würde dies bedeuten, dass die Tokens, welche Benutzer mit einer kleiner Bandbreite (*bwith*) wie eine Modemanbindung verdient haben, beim Eintausch beim Tokenausgabeserver dementsprechend gewichtet werden und dem Benutzer so mehr Tokens gutgeschrieben werden.

$$bias = \frac{1}{\sqrt{bwith}}$$

Dabei darf die Bandbreite nicht linear in die Rechnung einfliessen, da sonst die Zahl der verdienten Tokens kaum mehr von der Bandebreite abhängt, sondern nur noch von der Zeit, was die Besitzer von Breitbandanschlüssen benachteiligen würde.

Diese Gewichtung mutet in einem gewissen Masse unfair an, dass die Bandbreite nicht linear behandelt wird, andererseits haben Peers mit grösseren Bandbreiten in der Regel stets genügend Tokens. Der Vorteil davon ist, dass diese Massnahme zur Lastbalancierung beiträgt.

Zwar kann man argumentieren, dass diese Gewichtung bereits durch den Markt stattfindet, indem User mit geringerer Bandbreite diese als wertvoller einstufen und deshalb die Dateien teurer zum Download anbieten. In diesem Falle muss aber der Peer, welcher sich eine Datei herunterlädt dementsprechend mehr bezahlen und wird automatisch einen anderen Anbieter suchen, welcher die Dateien schneller und billiger anbietet. So werden Anbieter mit einer langsamen Internetanbindung indirekt gezwungen, den Content billiger anzubieten, als die Konkurrenz, da sonst die Customers stets einen schnelleren Provider Peer aussuchen würden, sofern diese nicht bereits ausgelastet sind.

Durch das ganze Verfahren ist es möglich, dass die Zahl der Tokens im System ansteigt und so zu einer Inflation führen kann, dieses Szenario wurde aber bereits zu Beginn des Kapitels besprochen.

3.3. Festlegung und Entwicklung des Preises einer Datei

In Pricing-Modellen, bei denen der Preis sich im Laufe der Zeit ändert, kann dies auf verschiedene Arten passieren:

- Die Preise werden von einer oder mehreren zentralen Stellen geregelt.
- Die Preise können nur vom Urheber der Datei festgelegt und verändert werden.
- Die Preise werden vom Angebot und der Nachfrage bestimmt und können vom Anbieter einer Datei bestimmt werden.

Im ersten Fall ist das Netzwerk von einer oder mehreren zentralen Stellen abhängig, welche die Preise und den Content kontrollieren. Diese Abhängigkeit schadet der Stabilität des Systems, da beim Ausfall dieser Stellen das ganze Netzwerk beeinträchtigt wird. Zudem ist es nötig, dass eine solche Stelle alle Dateien begutachten und bewerten muss, was mit einem grossen Aufwand verbunden ist. Zudem ist der Wert einer Datei subjektiv, so dass es durchaus vorkommen kann, dass die Bewertungsstellen gewisse Dateitypen unfair behandeln, da sie deren innere Werte nicht kennen. Sobald einmal ein Preis festgelegt ist, gilt dieser Preis für alle Peers, so dass diese praktisch keine Chance mehr haben, sich als Datenquelle attraktiver zu machen. Dies schadet vor allem den Peers mit einer langsamen Internetanbindung, da dort kaum jemand Dateien beziehen will.

Im zweiten Fall, wenn der Urheber den Preis der Datei bestimmen kann, stellt sich dasselbe Problem, wie bei Dateien mit unveränderlichen Preisen. In diesem Fall ist es möglich, dass manche Peers eine Datei zu einem tieferen Preis zum Download freizugeben wollen, damit sie ihre Uploadbandbreite besser ausschöpfen können. Wenn das Verändern des Preises aber nicht möglich ist, kann es vorkommen, dass diese Peers Plagiate von diesen Dateien erstellen, damit sie dann die Urheber dieser Kopien sind und deshalb den Preis festlegen können. Diese Dateien sind dann genau so stark verändert, dass sie gerade nicht mehr als identisch mit der Ursprungsdatei gelten. Je nach Netzwerk kann dies eine kleine Veränderung der Datei selber sein oder aber

nur ein anderer Name. Da in diesem Falle Dateien in verschiedenen Varianten auftauchen, wird es schnell unübersichtlich. So wird man bei einer Dateisuche Hunderte praktisch identische Files finden, von denen man aber nicht sicher sagen kann, ob sie von guter Qualität oder nur Fakes sind, da die Anzahl Quellen bei allen relativ klein ist und deshalb nicht mehr so viel über die Qualität aussagt.

Im dritten Fall wird der Preis der Datei nicht primär von dessen Wert bestimmt, sondern vielmehr vom Aufwand, diese Datei bereitzustellen und upzuloaden. So ist es auch nicht möglich, den Preis einzelner Dateien künstlich hochzuhalten. Jeder Einzelne kann selber bestimmen, wie viel er mindestens für eine bestimmte Datei haben will. Der Preis einer Datei einer Datei für einen Download passt sich dynamisch der Situation an und ist hauptsächlich von folgenden Einflüssen abhängig:

- Beanspruchung der Internetanbindung der Provider Peers
- Anzahl der Quellen einer Datei
- Nachfrage der Datei
- Eigene Strategien der Anbieter

Wenn der Benutzer die Freiheit hat, selber einen Preis für eine Datei festzulegen, wird dieser tendenziell solche Dateien anbieten wollen, die noch nicht so weit verbreitet sind aber trotzdem eine hohe Nachfrage aufweisen. Bei diesen Dateien ist die Chance, einen Upload zu verkaufen, grösser, und man kann dementsprechend etwas mehr für die Datei verlangen. Dadurch werden seltenere Dateien bevorzugt behandelt und im Netzwerk repliziert, was zur Folge hat, dass sich die Datei verbreitet. Es baut sich wieder ein Konkurrenzkampf auf, so dass dessen Preise wieder auf ein ähnliches Niveau fallen, wie die anderen Dateien im Netzwerk. Dadurch wird mit der Zeit ein Gleichgewicht entstehen, bei welchem das Verhältnis der Anzahl Anbieter (*Prov*), welche eine bestimmte Datei zur Verfügung stellen, und der Anzahl Nutzer (*Cust*) in einem bestimmten Zeitraum, welche das File beschaffen möchten, etwa konstant ist.

$$\frac{Cust}{Prov} = const$$

Ist eine Datei beliebt, werden sie automatisch auch mehr Leute anbieten. Wenn nach einer gewissen Zeit die Datei an Popularität verliert, sinkt die Nachfrage. Der Preis wird hingegen nicht zusammenfallen, da dadurch auch der Profit sinken würde. In diesem Falle wird einfach diese Datei im Verhältnis zum anderen Content weniger oft heruntergeladen, und verschwindet dadurch nach und nach von den Peers.

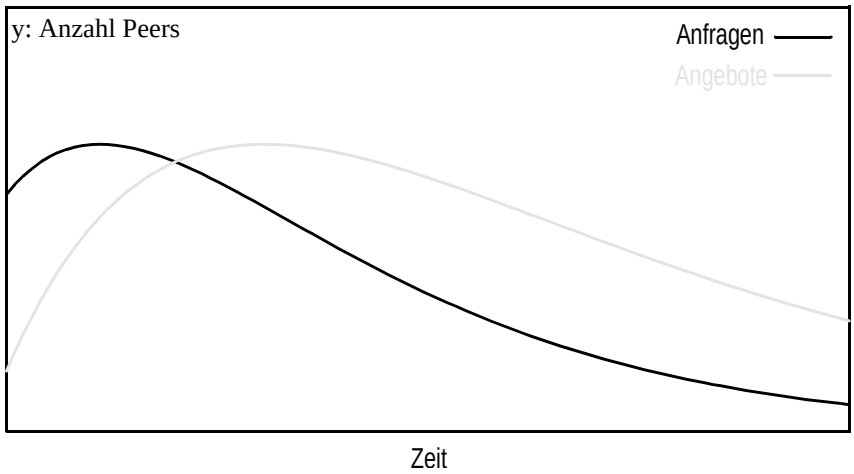

Abb 6: Da sich mit Downloads Tokens verdienen lassen, wird sich das Angebot der Nachfrage anpassen.

Da die verschiedenen Dateien als Datenmenge äquivalent sind, werden sie bei eingependeltem Verhältnis von Angebot und Nachfrage proportional zur Grösse (*Size*) etwa gleich viel kosten. Ein Preiszerfall wird in diesem Falle eher alle als einzelne Dateien betreffen.

$$Price = min(1, \frac{Cust}{Prov}) \cdot Size$$

3.3.1. Neuer Content im Netzwerk

Ein spezieller Fall tritt ein, wenn ein Nutzer eine neue Datei im Netzwerk bereitstellt. In diesem Falle ist er der einzige Anbieter und kann den Preis beliebig hoch ansetzen, da er keine Konkurrenz hat. Das Ganze hat aber einige Unsicherheiten: Der Provider Peer dieser Datei kann nicht beweisen, dass der Inhalt dieser Datei von guter Qualität ist. Da niemand das Risiko eingehen will, eine neue Datei, welche möglicherweise nur Datenmüll enthalten könnte, zu einem hohen Preis herunterzuladen, ist das Vertrauen der Nutzer gegenüber diesem Anbieter oder die Vertrauenswürdigkeit des Anbieters im Netz ausschlaggebend, ob die Datei einen Kunden findet, oder ob die User erst einmal vorsichtig sind. Falls man den Anbieter schon von anderen erfolgreichen Downloads kennt, ist die Chance besser, dass er eine Datei zu einem hohen Preis verkaufen kann. Hat jedoch der Anbieter schon einmal das Vertrauen der User missbraucht, werden diese Benutzer diesen Anbieter meiden und vielleicht sogar die anderen Benutzer warnen. Dies kann mit Hilfe von Ratings geschehen [13], oder mit virtuellen Dateinamen, welche bei einer Suchanfrage auch aufgelistet werden und neben dem Dateititel auch noch die Warnung enthalten.

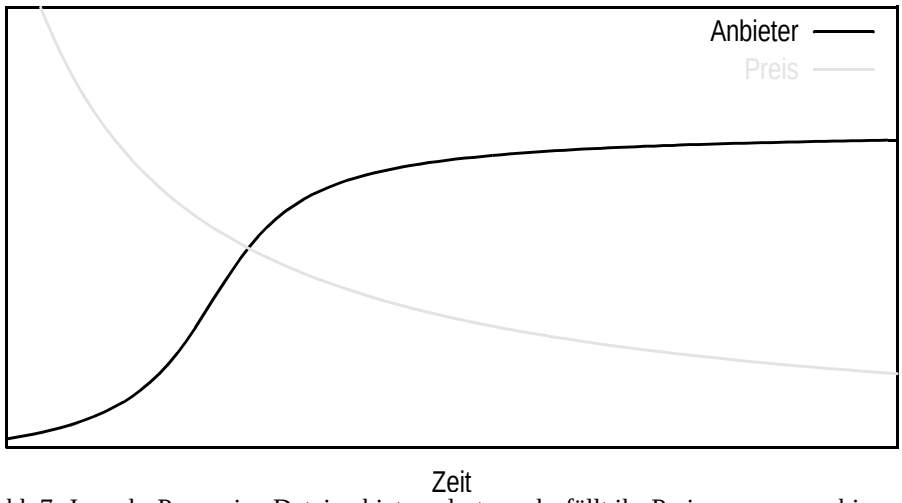

Abb 7: Je mehr Peers eine Datei anbieten, desto mehr fällt ihr Preis zusammen, bis er sich auf einem konstanten Wert einpendelt.

Bei gewissen streamingbasierten Dateitypen vie Audio- oder Videofiles ist es jedoch möglich, dass die anderen Benutzer einmal vorsichtig einen kleinen Teil der ganzen Datei herunterladen und diesen erst einmal begutachten. Wenn es sich um Datenmüll handelt, kann der Download immer noch abgebrochen werden. Ist jedoch der Anfang von guter Qualität, stehen die Chancen jedoch gut, dass dies beim Rest auch so bleibt.

Falls nun eine Datei von guter Qualität einen Abnehmer fand, so wird dieser wahrscheinlich diese Datei auch anbieten wollen, weil sich damit viele Tokens verdienen lassen. Damit die Leute aber von ihm runterladen und nicht vom ersteren Anbieter, muss er den Preis unterbieten. In diesem Falle ist aber der erstere Anbieter gezwungen, auch mitzuziehen, und so werden sich wahrscheinlich die Preise so stark absenken, bis die Bandbreite von beiden aufgefüllt ist. Trotzdem zerfällt der Initialisierungspreis (P_{init}) nicht umgekehrt proportional zur Anzahl Anbieter (*Prov*), da die Zahl der Interessenten bei sinkendem Preis zunimmt und es deshalb keinen Grund gibt, die Preise zu weit zu senken. Will man diesen Effekt mathematisch visualisieren, so hat die Anzahl Quellen einen Einfluss auf den Preis einer Datei, aber weniger als linear. man könnte den Preis (*Preis*) einer Datei mit folgender Formel approximieren:

$$Preis = \frac{P_{init}}{\sqrt{Prov}}$$

Je nach Strategie dieser Anbieter kann es aber auch vorkommen, dass sie die Preise nicht so stark senken, aber dafür mehr Bandbreite für andere Uploads übrig haben. In diesem Fall wird der Preiszerfall erst später eintreten, wenn die Datei sich etwas stärker verbreitet hat. Hier spielt die Taktik der Anbieter in einem grossen Masse hinein.

Es kann aber auch sein, dass, wenn jemand als einzige Person eine Datei zum Download anbietet, dieser versucht, möglichst viele Kunden auf einmal zu finden. In diesem Falle ist er noch der einzige Anbieter, und kann bei allen Downloads den Preis hoch halten. So wird er wahrscheinlich gegen Schluss des Downloads diesen so lange wie möglich verzögern, damit er nicht allzu früh Konkurrenz erhält. Dieses Szenario lässt sich kaum verhindern, aber man kann ähnlich wie bei schlechtem Content mittels Ratings die anderen Benutzer warnen.

3.3.2. Einfluss des Urheberrechts auf den Preis einer Datei

Wenn User in einem Peer-to-Peer Netzwerk urheberrechtlich geschützte Dateien zum Download anbieten, begeben sich diese auf rechtliches Glatteis. Je nach dem, wie viel Content bestimmte Anbieter zur Verfügung stellen, lohnt es sich, diese zu identifizieren und zur Rechenschaft zu ziehen. Aus diesem Grunde, werden einige Nutzer eher vorsichtig mit urheberrechtlich geschütztem Material sein und hauptsächlich ungeschützten Content anbieten.

Aus diesem Grunde werden die ungeschützen Dateien im File-sharing Netzwerk eine grössere Verbreitung haben, als die Copyrightgeschützten. Da aber die meisten Benutzer alle Typen von Dateien suchen, wird die Nachfrage bei den urheberrechtlich geschützten Materialien auf einem Provider Peer grösser sein, als bei freiem Content, was sich wahrscheinlich auf den Preis niederschlägt.

Der Effekt, dass sich urheberrechtlich geschütztes Material zu einem besseren Preis verkaufen lässt, kann einzelne Peers dazu verleiten, nur noch solches Material anzubieten. Da viele Nutzer vorsichtig sind mit urheberrechtlich geschütztem Material, könnte dies die Lastbalancierung des Netzwerkes beeinträchtigen, weil in diesem Falle nicht mehr so viele Provider Peers übrig bleiben.

3.3.3. Anbieten von Dateien ohne Gegenleistung

Wie es schon in früheren File-sharing Netzwerken Peers gab, welche uneigennützig Dateien zum download anbieten, ohne selber viel herunterzuladen, wird es wahrscheinlich auch in Tokenbasierten Netzwerken Provider Peers geben, welche Dateien zum Download anbieten, ohne dass sie dafür eine Gegenleistung erwarten. Dies würde heissen, dass diese Peers die Dateien für null Tokens zum Download offerieren.

Es ist offensichtlich, dass diese Peers bei den Customer Peers dementsprechend beliebt sind, und die Uploadbandbreite stets voll ausgelastet ist. Die natürliche Folge davon wäre das Bilden einer Warteschlange, in welcher sich die Customer Peers einreihen. Zudem ist es möglich, dass einzelne Customer Peers dem Anbieter trotzdem einen kleinen Betrag grösser Null offerieren, damit sie vorne in der Warteschlange eingereiht werden.

Nichtsdestotrotz werden diese Peers den Markt zu einem gewissen Grade sättigen und dabei Customer Peers davon abhalten, Tokens in den Umlauf zu setzen. Dies kann den Peers, welche Content anbieten, um Tokens für eigene Downloads zu verdienen, das Leben schwer machen, da sie mehr Mühe haben, die eigenen Dateien unter die Leute zu bringen und so beinahe gezwungen werden, sich von Gratisanbietern zu bedienen. Als Gegenmassnahme könnten diese Peers ihre Preise so stark zu senken, bis genügend Interessenten sich dafür entscheiden diesen Betrag zu bezahlen anstatt lange in einer Warteschlange anzustehen.

Auf der anderen Seite nützen diese altruistischen Peers dem Netzwerk insofern, als dass die Anzahl der angebotenen Dateien ansteigt. Dadurch wird das Peer-to-Peer Netzwerk beliebter, und die Benutzerzahl wird ansteigen.

3.4. Bid-Ask Auction Methode als Pricing Modell für P2P Filesharing

Pricing Modelle für Peer-to-Peer Netzwerke können in zwei verschiedene Kategorien aufgeteilt werden:

- Pricing Modelle, bei denen der Preis im Laufe der Zeit fix bleibt
- Pricing Modelle, bei denen der Preis sich im Laufe der Zeit ändern kann

Falls der Preis einer Datei fix bleibt, vereinfacht sich einiges im ganzen Peer-to-Peer Netzwerk. Die Preise müssen nur einmal synchronisiert werden, und preismässig kommt es nicht darauf an, wo man seinen Content bezieht. Dieser Zustand beschneidet jedoch die Freiheit der einzelnen Nutzer, indem ihnen vorgegeben wird, zu welchem Preis sie Datei zu verkaufen haben. Die Folge davon habe ich schon im Kapitel [3.3] genauer beschrieben. Da sich bei dem vorliegenden Projekt die Preise der Dateien mit der Zeit ändern können, gehe ich nicht weiter auf Netzwerke mit fixen Preisen ein.

Wenn sich die Preise permanent ändern, müssen diese mit Hilfe eines Preisverteilungsmechanismus auf den verschiedenen Peers stets nachgeführt werden. Entweder müssen Preise in diesem Falle dementsprechend oft aktualisiert und wieder verteilt werden (Push-Methode). Alternativ kann ein Customer Peer, welcher nach einem bestimmten File sucht, die Provider Peers, die das File im Angebot haben, immer wieder anfragen, ob der Preis noch gültig ist, oder sich dieser inzwischen verändert hat (Pull-Methode). In diesem Falle werden nur diese Preisinformationen permanent neu verteilt, welche wirklich von Interesse sind. Ein Customer Peer, der an einem bestimmten File interessiert ist, kann nun diesen Vorgang solange wiederholen, bis der Preis z.B. auf ein Niveau gesunken ist, bei welchem sich der Download für ihn lohnt.

Um den vielen Anfragen vorzubeugen, welche daraus resultieren würden, bietet sich ein dritter Ansatz an, welcher in einer ähnlichen Form bereits an Börse verwendet wird: Ein Customer Peer gibt den jeweiligen Provider Peers an, zu welchem Preis er bereit wäre, ein bestimmtes File herunterzuladen (Bid/Ask-Methode). Da der vom Customer Peer angebotene Preis in der Regel kleiner ist als, die Offerte des Provider Peers, kommt der Download wahrscheinlich nicht sofort zu Stande. Der Provider Peer nimmt aber dieses Angebot zur Kenntnis und speichert es in einer Liste. Sobald jedoch ein Zeitpunkt auftritt, bei welcher z.B. die Uploadbandbreite des Provider Peers nicht genügend ausgelastet ist und daher der Preis für das entsprechende File sinkt, kann sich der Provider Peer die Liste mit den Downloadangeboten anschauen, und dasjenige Angebot aussuchen, welches den höchsten Profit erbringt. Somit kontaktiert der Provider Peer den entsprechenden Customer Peer, und fragt nach, ob er noch immer dazu bereit wäre, das entsprechende File zu dem angebotenen Preis herunterzuladen. Falls dies noch der Fall ist, wird der Download eingeleitet; andernfalls wird das Downloadangebot von der Liste gelöscht und das nächstbeste Angebot ausgewählt. Diese Liste hat also die Funktion einer Warteschlange, bei der die Angebote nicht nach Wartedauer geordnet sind, sondern nach dem Profit pro Zeiteinheit, welche dieser Download dem Provider Peer bringt. In der Regel ist dies das Verhältnis zwischen dem Preis (P) und der Dateigrösse (S), wenn nicht der Preis bei bestimmten Dateien künstlich hochgehalten wird.

$$Priority = \frac{P}{S}$$

3.4.1. Management der Warteschlange

Damit sich die Anzahl der Downloadangebote auf einem Server nicht ins Unermessliche steigert, ist es sinnvoll, die Angebote auf eine beschränkte Gültigkeitsdauer zu beschränken. Zudem ist es möglich, dass ein Customer Peer von sich aus ein Downloadangebot ändert oder

zurückzieht. Ob er dies dann wirklich dem Provider Peer meldet, ist allerdings fraglich, da dem Customer Peer keine Nachteile dadurch entstehen, wenn das Angebot auf dem Provider Peer liegen bleibt, auch wenn er nicht mehr an dem File interessiert ist.

Wenn neue Downloadangebote beim Provider Peer ankommen, ist es von Vorteil, wenn er diese mit den schon bestehenden vergleicht. So ist es zum Beispiel vorteilhaft, die Anzahl Downloadangebote von ein und demselben Peer zu beschränken, damit dieser nicht die ganze Warteschlage auffüllen kann. In diesem Falle kann einer möglichen Attacke vorgebeugt werden: Wenn ein Benutzer die Möglichkeit hat, beliebig viele Downloadangebote bei einem Provider Peer abzusetzen, kann dieser gefälschte Angebote beim Provider Peer einreichen, die besser sind, als alle anderen bisherigen Kandidaten in der Warteschlange. Wenn er nun genug viele davon einreicht, wird der Provider Peer die anderen aus der Warteschlange verdängen, weil nicht mehr genug Platz darin ist, bis nur noch dieser Anbieter übrig bleibt. Wenn dann einmal die Downloads bei diesem Provider abnehmen, wird dieser in der Warteschlange nachschauen und den Peer mit den gefakten Downloadangeboten kontaktieren. Da er natürlich nicht bereit ist, zu diesem Preis eine Datei herunterzuladen, muss der Providerpeer nach und nach alle Angebote von diesem Peer löschen, so dass keine Angebote übrig bleiben. Sobald die Warteschlange leer ist, wird er dann bereit sein, zu einem schlechteren Preis eine Datei abzugeben, als bei den Angeboten, welche in der Warteschlange waren, bevor der Angreifer seine Angebote abgegeben hat. Da die Priorität in der Warteschlange nicht von der Wartedauer abhängt, sondern vom Profit für den Anbieter, wird sich die Reihenfolge der Downloadangebote kaum ändern. Es ist nicht sehr wahrscheinlich, dass neue Downloadangebote stets schlechter sind, als die bereits existierenden; aus diesem werden sich diese irgendwo in der Mitte oder gar vorne einordnen. So wird es wahrscheinlich einen Punkt in der Warteschlaufe geben, hinter welchem die Downloadangebote wahrscheinlich nie die Chance mehr haben, an die Reihe zu kommen. Vor diesem Punkt nimmt die Wahrscheinlichkeit nach und nach zu, bedingt durch die Schwankungen während des Tages und der Woche. Es bringt dasher nichts, die Warteschlange beliebig gross zu wählen.

Es ist also nötig, regelmässig die Warteschlange durchzugehen und aufzuräumen. Wenn das Gültigkeitsdatum eines Downloadangebots abgelaufen ist, ist der Fall klar, dann kann es sofort gelöscht werden. Bei den anderen lohnt sich vielleicht einmal eine Rückfrage, um zu schauen, ob dieser Rechner noch immer an dem Download interessiert ist, oder ob er überhaupt noch erreichbar ist. Falls letzteres nicht der Fall ist, wäre es nicht gut, wenn man das Downloadangebot schon nach dem ersten Kontaktierungsversuch aus der Warteschlange löscht, denn es kann immer sein, dass Peers für eine gewisse Zeit offline sind, oder dass Anfragen im Internet verloren gehen. Unter Umständen kann es auch sinnvoll sein, Benutzer, welche aus der Warteschlange gefallen sind, zu benachrichtigen, damit sie die Möglichkeit haben, ein besseres Angebot einzureichen.

Wenn der Customer Peer zu dem Zeitpunkt, zu dem der Provider Peer ihm einen Download anbietet, gerade offline ist, geht er das Risiko ein, dass er die Gelegenheit nicht nutzen kann, das File zu holen. In diesem Falle sollte aber der Provider das Downloadangebot nicht von der Liste löschen, da der Customer Peer sich nach kurzer Zeit wieder ins Netz einbinden kann. Dabei sollte er auch dann wieder als derselbe Customer Peer identifizierbar sein, auch wenn er eine andere IP erhalten hat. Falls ein Customer Peer permanent von der Bildfläche verschwindet, könnten die Downloadangebote nach Ablauf der Gültigkeitsdauer oder nach einer bestimmten Anzahl Verbindungsversuchen von seiten des Provider Peers automatisch gelöscht werden.

Falls der Uploadlink eines Provider Peers mit volumenabhängigen Kosten belastet ist, ist es möglich, dass sich zu tiefe Downloadangebote für den Server überhaupt nie lohnen, was bedeutet, dass es für den Provider Peer mehr Profit bringt, die Bandbreite nicht vollständig auszulasten, anstatt die Dateien zu unrentablen Preisen anzubieten. In diesem Falle können entsprechende Angebote auch direkt abgelehnt und gelöscht werden.

3.4.2. Informationspolitik

Wenn der Preis für ein bestimmtes File noch nicht unterschritten ist, muss der Customer Peer für unbestimmte Zeit warten, bis der Provider Peer ihm einen Download offeriert. Je nach Angebot kann dies aber lange dauern, da bei einem schlechten Angebot dieses Customer Peers immer wieder andere Downloadangebote an den Provider Peer gerichtet werden, die für ihn einen besseren Profit bringen, und deshalb weiter vorne in der Warteschlange eingereiht werden. Diese Zeit kann für den Customer Peer recht mühsam sein, da er nie weiss, wie lange er noch warten muss, vor allem, wenn er nicht allzu lange warten kann. Es ist natürlich möglich, dass dieser Peer das Downloadangebot gleichzeitig an verschiedene Provider Peers richtet. Dies werden aber die meisten Customer Peers tun, wenn sie auf der Suche nach dem gleichen File sind. Es herrscht also generell eine grosse Unsicherheit, weil die verschiedenen Benutzer nicht wissen, an welcher Position sie sich in der Warteschlange befinden und wie lange es in etwa dauern wird, bis sie an die Reihe kommen.

Die Provider Peers haben im Prinzip die Möglichkeit, den Customer Peers diese Information zu geben. Nun stellt sich aber die Frage: Bringt dies dem Provider Peer etwas oder ist es taktisch intelligenter, diese Information geheim zu halten?

Auf der einen Seite kann man argumentieren, dass die Customer Peers vielleicht bereit sind mehr Tokens für einen bestimmten Download auszugeben, damit sie nicht so lange über den Zustand im Dunkeln gelassen werden. In diesem Falle kann der Provider Peer dieses Wissen als Machtmittel benutzen, und dem Customer nicht sagen, für wie viele Token man in welcher Zeit man einen Download erreichen kann, damit die Customer mehr bezahlen, als wirklich nötig.
Auf der anderen Seite wird es wahrscheinlicher, dass die Customer einfach aufgeben, das File zu holen, nachdem sie schon eine gewisse Zeit erfolglos gewartet haben. Kurzfristig könnte es dem Provider Vorteile bringen, diese Informationen geheim zu halten. Für die Customer ist aber diese Situation nicht optimal, und könnte sie zu einem gewissen Grade entmutigen, im Netzwerk Dateien zu tauschen. Auch hätte es der Provider Peer sicher lieber, wenn ihm diese Information gegeben würde, wenn er einmal eine Datei sucht. Man will ja schliesslich nicht die Leute vom Filesharing Netzwerk abschrecken, sondern sucht möglichst viele Teilnehmer, damit das Netzwerk möglichst optimal funktioniert.

Wenn nun die Customer die Information über ihre Position in der Warteschlange und die erwartete Wartezeit erhalten, kann es durchaus sein, dass sie ihr Angebot noch etwas verbessern und mehr für den Download anbieten, um schneller an die Reihe zu kommen. Weiter werden sie tendenziell nicht so viele Downloadofferten an die verschiedenen Provider Peers verteilen, wenn sie ungefähr wissen, bei welchen sie schneller an die Reihe kommen. Dies hilft die Zahl der Einträge in den Warteschlangen zu senken, die nie zu einem Download führen werden. Ich würde meinen, dass es dem Netzwerk als Ganzes nützt, wenn die verschiedenen Peers grosszügig mit solchen Informationen umgehen, damit die verschiedenen Peers wissen, woran sie sind, und so die Benutzerfreundlichkeit des Systems steigern.

Nichtsdestotrotz können die Provider Peers nicht immer zuverlässige Informationen abliefern. Sie können zum Beispiel einem Customer Peer mitteilen, an welcher Position er gerade in der Warteschlange steht, daraus eine Vorhersage abzuleiten, wann er wirklich an die Reihe kommen wird, ist äusserst schwierig. Zwar kann der Provider anhand von Statistiken aus der Vergangenheit eine durchschnittliche Wartezeit errechnen, diese kann aber unter Umständen falsch sein, wenn überraschenderweise viele neue Downloadofferten eintreffen. In diesem Falle wäre es aber möglich, dass die Customer Peers nach einer gewissen Zeit nochmals eine Anfrage nach der erwarteten Wartezeit stellen.

3.4.3. Anteil von direkten Downloads und Downloadofferten

Wenn sich das Anbieten von Downloads durchsetzt, könnte sich der Markt im Laufe der Zeit so verändern, dass seltener Dateien direkt heruntergeladen werden, sondern immer beim Server in eine Warteschlange der Downloadofferten gesetzt werden. In dieser können Peers die Position darin durch das Preisangebot selber steuern.

Es ist nun möglich, dass die Customer Peers nicht mehr die Dateien sofort herunterladen und dabei den Preis bezahlen, welcher auf der Preisliste steht, sondern das Downloadangebot so wählen, dass sie genau an die Spitze des Warteschlange gesetzt werden, ohne viel mehr als nötig auszugeben. Wenn dies natürlich alle machen, werden sich diese Downloadofferten wieder nach hinten verschieben, und die Preise sich immer mehr den Listenpreisen annähern, also den Preisen, welche die Provider Peers bereits für die entsprechenden Dateien reserviert haben. Inwiefern das dann wirklich passiert, hängt von der Nachfrage im Netzwerk generell und von der Bandbreite des Provider Peers ab. Wenn die Nachfrage nach Dateien sehr gross ist, könnte es auch sein, dass die Preise in die Region kommen, in welcher auch die Listenpreise liegen. In diesem Falle wird der Provider Peer diese aber nach oben korrigieren damit sie nie kleiner werden, als diejenigen in der Liste der Downloadangebote. Auch sonst wird die Diskrepanz zwischen den offiziellen Preisen der Dateien und denjenigen in den Downloadofferten nicht so gross sein, wenn die Peers mässig ausgelastet sind, da die Peers die Listenpreise auch von Zeit zu Zeit dem Markt anpassen.

Vorteile haben hier Peers, die meist nur Dateien zum Download anbieten, aber selten etwas herunterladen. Diese haben nämlich genügend Tokens, und können sich so stets an die Spitze der Liste setzen oder die Dateien direkt beziehen.

3.5. Das Peer-to-Peer Netzwerk

Ein Peer-to-Peer Netzwerk setzt sich aus Computern zusammen, die verschiedene Rollen einnehmen können. Je nach Verhalten dieser Peers und deren Häufigkeit verändert sich die Charakteristik des ganzen Verbundes. Die wichtigsten Typen von Peers mit ihren Eigenschaften werden hier einmal kurz vorgestellt.

3.5.1. Die verschiedenen Typen von Peers

Die verschiedenen Peers im P2P Netzwerk können verschiedene Rollen einnehmen:

- Reine Server, die ausschliesslich Dateien anbieten
- Peers, welche nur gerade soviel für den Download bereitstellen, damit sie genügend Tokens für die eigenen Downloads haben.
- Reine Clients, welche nur Dateien herunterladen

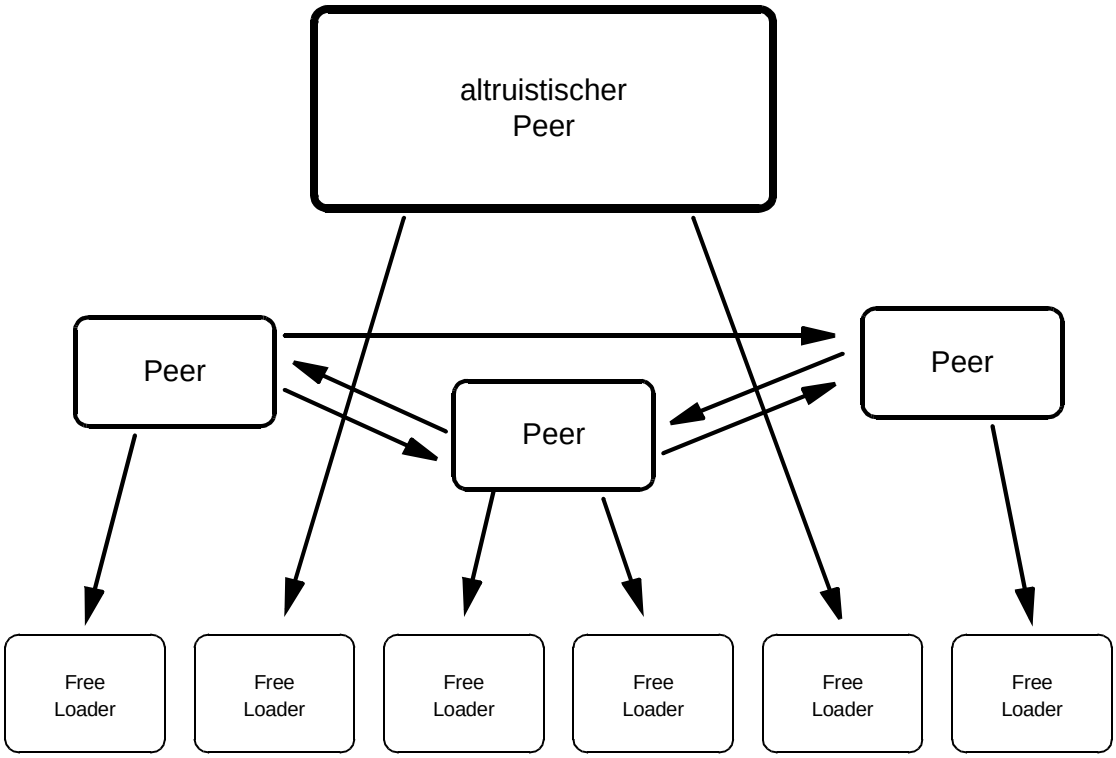

Abb 8: ein verbreiteter Aufbau eines Peer-to-Peer Netzwerks: Wenige Peers, delche nur Content anbieten, einige mehr, wessen handelsbilanz in etwa neutral ist und eine grosse Menge von Teilnehmern, welche nur profitieren.

Peers, welche als Server agieren

Diese erste Gruppe kann wiederum in zwei Kategorien eingeteilt werden:

- Server, die uneigennützig möglichst viele Dateien ins Netz verteilen wollen
- Server die als Hauptzweck haben, den Profit zu maximieren, also mit den Downloads so viel zu verdienen, wie möglich.

Auch wenn eine Währung als Gegenwert zu den Dateien im Peer-to-Peer Netzwerk eingesetzt wird, wird es sicher altruistische Peers geben, welche ihren Content zur Verfügung zu stellen, ohne dabei die Absicht zu haben, gross Gewinn zu machen. Manchmal werden diese Peers ihren Content sehr günstig anbieten, teilweise sogar gratis. Warum es solche Server gibt, kann man nicht definitiv erklären, ich nehme jedoch an, diese werden von solchen Personen am Laufen gehalten werden, dessen Lebensphilosophie es ist, dass Informationen allen Menschen gehört und deshalb verteilt werden muss. Vielleicht ist es eine Art kommunistische Ansicht des Lebens, in welcher es nicht gut ist, dass einige Menschen gegenüber anderen a priori Vorteile haben nur weil sie bestimmte Informationen besitzen, welche andere nicht haben. Es könnte aber auch der Gedanke sein, dass sie von anderen Leuten auch schon gratis Content erhalten haben und deshalb aus Dankbarkeit sich gleich verhalten. In vielen Fällen stellen solche Server eine grosse Zahl von verschiedenen exotischen Dateien bereit, um die kulturelle Vielfalt im Peer-to-Peer Netzwerk zu gewährleisten. So findet man an solchen Orten Content, welcher ohne diese Netzwerk Gefahr laufen würde, für immer in Vergessenheit zu geraten.

Auch wenn man die Absicht dieser Personen nicht vollkommen verstehen kann, bilden sie trotzdem einen wichtigen Eckpfeiler im Peer-to-Peer Netzwerk. Zum einen vergrössern sie mit der eigenen Kapazität den Gesamtdurchsatz des Verbundes, zum anderen könnten die durch ihren Content, den sie gratis oder sehr billig zur Verfügung stellen, eine Konkurrenz für die anderen Peers darstellen, und so die Preise drücken. Die meisten Benutzer haben Interesse, Content möglichst günstig aus dem Netzwerk zu beziehen, und werden natürlich diese Server bevorzugen. Je nach dem wie vielfältig der Content und wie gross die Uploadbandbreite der uneigennützigen Server im Verhältnis zu allen Downloads ist, desto stärker ist dieser Preiskampf spürbar. Falls diese Provider Peers ziemlich den ganzen Markt abdecken können, wird es für die anderen Peers relativ schwierig, selber noch Tokens zu verdienen. In diesem Falle ist es jedoch nicht mehr so wichtig, viele Tokens zu verdienen, da man nicht mehr so viele für die eignen Downloads benötigt. Das Gleichgewicht der Tokens sollte zu diesem Zeitpunkt gegeben sein. Wenn die altruistischen Server eine zu grosse Bandbreite haben, funktioniert die Lastbalancierung nicht mehr so gut, wenn alle von diesen wenigen Servern herunterladen. Dies ist aber auch nicht nötig, da ja die Bandbreite zu diesem Zeitpunkt ausreicht.

Sobald jemand mit der Downloadgeschwindigkeit nicht zufrieden ist, wird er sich überlegen, doch ein paar Tokens zu bezahlen um eine schnellere Datenquelle zu ergattern.

Die andere Gruppe von Peers, welche hauptsächlich als Server agieren, bieten den Content nicht gratis an, sondern verlangen dafür gerade so viel, dass die Uploadbandbreite gerade noch ausgelastet wird. Die Idee dahinter ist die volle Bandbreite auszunutzen, aber dabei trotzdem so viel an einer Datei zu verdienen wie möglich. Wenn jedoch die altruistischen Server praktisch die ganze Nachfrage abdecken können, werden diejenigen Server, die an den Uploads verdienen wollen, zurückkrebsen müssen und die Preise entsprechend senken.

Das Verhalten dieser Server wird von der Frage beeinflusst, ob man im File-sharing Netzwerk die verdienten Tokens gegen echtes Geld eintauschen kann. Falls das der Fall ist, wird der Trend zu diesem Typ von Servern viel stärker sein, als wenn man die Tokens nur innerhalb des Netzwerkverbundes als Zahlungsmittel für Dateien nutzen kann. Wenn zu viele Server die Tokens im Netzwerk einsammeln um sie dann der Tokenausgabestelle zu verkaufen, wird der Wert der Tokens gemessen an realem Geld abnehmen. Dadurch sinkt natürlich die Menge Tokens im ganzen Verbund und die Tokenausgabeserver müssen dementsprechend reagieren (Siehe Kapitel [3.2]). Eine andere Variante ist, dass nun die Provider Peers, welche viele Tokens gesammelt haben, diese direkt an andere Peers gegen richtiges Geld verkaufen, damit die Tokenausgabestellen keine Marge auf die Tokens legen kann, bevor sie sie an die Benutzer weiterverkauft.

Je nach dem, ob in naher Zukunft die durchschnittliche Bandbreite der Benutzer oder der Traffic im Peer-to-Peer Netzwerk schneller anwächst, werden diese zwei Typen von Servern die Nachfrage mehr oder weniger gut abdecken können, was dann einen direkten Einfluss auf die Preise im Verbund hat.

Neutrale Peers

Diese Gruppe macht wohl den grössten Teil der Nutzer aus. Sie haben nicht speziell Interesse, den anderen Benutzern viel Content anzubieten, sondern machen dies nur, damit sie genügend Tokens verdienen können, um die eigenen Downloads finanzieren zu können. Dieses Veralten kann auf zwei verschiedene Arten erklärt werden: Zum einen ist es möglich, dass ihre Internetanbindung nur ein bestimmtes Freivolumen hat, so dass sie draufzahlen müssen, wenn die ins Netz gesendete Datenmenge eine bestimmte Grenze überschreitet. Zum anderen ist auch denkbar, dass sie unsicher über die gesetzliche Situation beim Filesharing sind, und sich deshalb so wenig wie möglich auf rechtliches Glatteis begeben wollen.

Oft haben diese Peers keine fixe IP und können sich beliebig wieder vom Internet disconnecten. Wenn andere Benutzer von diesen Peers Dateien herunterladen wollen, müssen sie damit rechnen, dass diese plötzlich von der Bildfläche verschwinden und auf unbestimmte Zeit nicht mehr auftauchen. In diesem Falle macht es Sinn, solch Dateien auszusuchen, welche als identische Kopie auf verschiedenen anderen Peers vorhanden sind, so dass man den Download auch wirklich beenden kann.

Da diese Peers nur eine gewisse Anzahl Tokens für die eigenen Downloads benötigen, beschränken sie sich in der Regel auf eine kleinere Anzahl Dateien, die vergleichsweise populär sind. So kann der Peer eher sicher sein, dass sich gleich ein paar Interessenten finden, wenn er mit dem Peer-to-Peer Netzwerk verbindet. Sind hingegen die Dateien eher exotisch, ist es möglich, dass er länger auf potentielle Kundschaft warten muss.

Nun kann man sich fragen, wie erfolgreich diese Peers sein werden, wenn sie schnell mal ein paar Token für die eigenen Downloads verdienen wollen. Dies hängt, wie schon im vorherigen Abschnitt beschrieben, von der Leistungsfähigkeit der Server ab, welche hauptsächlich Content anbieten. Beim Austausch von Dateien zwischen neutralen Peers kann man in etwa davon ausgehen, dass die Anzahl verdienter Tokens pro gesendete Datenmenge beim Upload und Download in der Regel gleich ist. Je nach dem, um wie viel schneller der Download bei der eigenen Internetverbindung ist als der Upload, muss ein Peer meist um ein vielfaches länger im Internet Content uploaden, als er effektiv beziehen kann. Da ein Client gleichzeitig Content hinauf- und herunterladen kann, kann dies alles gleichzeitig geschehen, und wenn dann der Peer sowieso schon an das Filesharing Netzwerk angeschlossen ist, spricht nichts dagegen, in dieser Zeit ein paar günstige, aber langsame Downloads im Hintergrund am Laufen zu haben.

Peers, welche nur Content beziehen

Peers, welche kein Content zur Verfügung stellen, haben auch keine Möglichkeit, Tokens im Netzwerk zu verdienen, bevor Rechenkapazität und andere Dienstleistungen mit Hilfe von Tokens abgerechnet werden.

In dieser Situation ist seine Auswahl an Content auf die altruistischen Server beschränkt. Diese erhalten dadurch natürlich dementsprechend viele Downloadanfragen von Peers, die keine Tokens bezahlen wollen, und er muss dementsprechend warten. Diese Situation tritt bereits schon heute bei vielen File-sharing Netzwerken auf, in welchen keine virtuelle Währung existiert.

Falls eine Möglichkeit eingeführt wird, Tokens gegen echtes Geld zu beziehen, können sie wieder aktiver am Tauschgeschehen teilnehmen. Es ist jedoch zu bezweifeln, dass alle bereit sein werden, echtes Geld für Tokens auszugeben.

3.5.2. Zentrale Stellen

Im Prinzip wäre es die Idee eines Peer-to-Peer Netzwerkes, dass dieses ohne jegliche zentrale Stellen funktionieren kann. In diesem Falle müssen sich die Peers untereinander selbst finden und eine Verbindung aufbauen. Leider ist nicht immer die IP Adresse eines Teilnehmers des Verbundes bekannt, so dass ohne zentrale Informationsquelle eine Verbindung nicht zustandekommen kann. Aus diesem Grund ist mindestens eine Stelle nötig, welche die Adressen einiger Peers im Netzwerk kennt, welche immer über die selbe Adresse erreichbar sind und daher als Einstiegspunkt für das Peer-to-Peer Netzwerk benutzt werden können.

Damit das ganze System am Laufen bleibt, sind je nach Netzwerk mehrere zentrale Stellen nötig:

- Server für den Einstieg ins Netzwerk
- Tokenausgabeserver
- Indexserver
- Supernodes

Bei kleineren Netzwerken genügt es, wenn jeder Peer die Adressen einiger anderer Peers kennt. Auf diese Weise wird ein Graph aufgespannt, welcher n-fach zusammenhängend ist, wobei n im Idealfall so gross ist, wie die Anzahl Nachbarn, die er in seiner Liste gespeichert hat. Hier braucht es nur einen Einstiegspunkt, von welchem man die IP eines Teilnehmers erfährt. Häufig geht dies mit Hilfe einer URL, damit diese bei einer Änderung der IP nachgeführt werden.

Leider skaliert dieses System nicht so gut, wie man es bei Gnutella schon gesehen hat. Je grösser der Zahl der Peers ist, desto mehr Bandbreite geht für Suchanfragen und Management des Systems verloren, weil die Knoten alle diese Messages weiterrouten müssen.

Indexserver

Eine Vereinfachung des Ganzen kann damit erzielt werden, indem jeder Peer einer zentralen Stelle seine Contentliste sendet, und diese zentrale Stelle dann die Suchanfragen auswertet. So nimmt der organisatorische Traffic im System ab; diese Indexserver müssen jedoch mit einer hohen Bandbreite ans Internet angebunden sein, um alle Anfragen bewältigen zu können. Diese Bandbreite begrenzt die Anzahl möglicher Teilnehmer im Netzwerk. Um diesen Engpass zu beseitigen, ist es möglich, mehrere Indexserver parallel zu betreiben, die sich dann untereinander die Arbeit aufteilen.

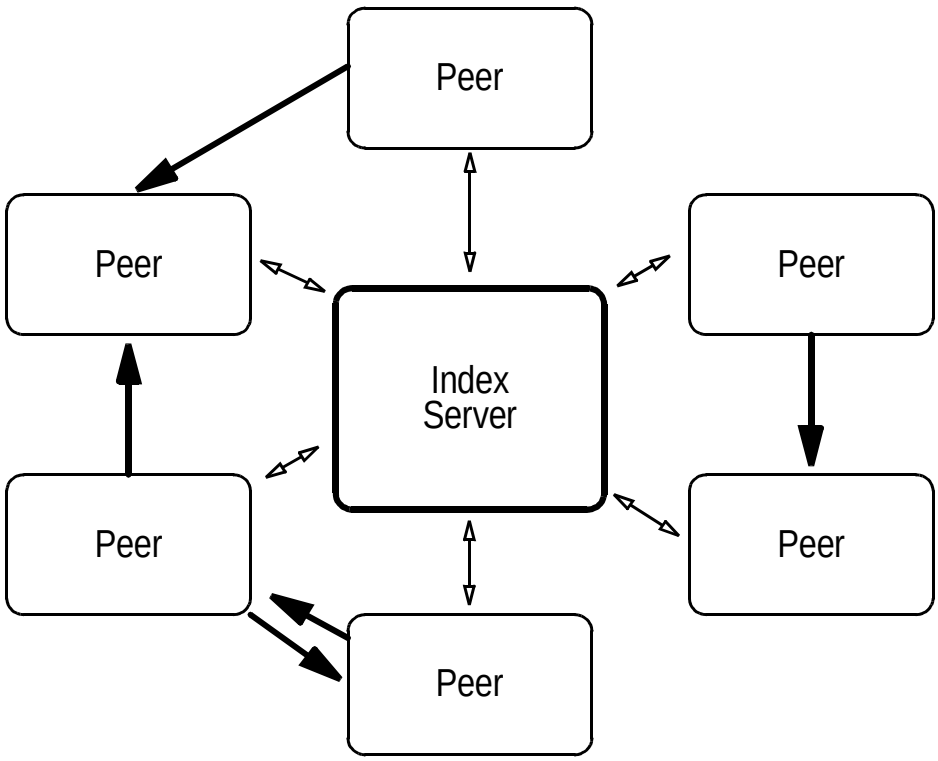

Abb 9: Die Indexserver beantworten Suchanfragen der Peers; am Dateiaustausch sind sie jedoch nicht beteiligt.

Die Indexserver müssen in regelmässigen Abständen die Informationen über den Content und dessen Preise aktualisieren, damit das Verzeichnis aktuell bleibt. Dies kann der Indexserver aktiv erledigen, indem er die einzelnen Peers in regelmässigen Abständen kontaktiert und sich die Preisinformationen holt. Der Server merkt bei jedem Aktualisierungszyklus, welche Peers nicht mehr online sind, und kann diese Einträge aus dem Verzeichnis löschen. Andererseits ist es auch möglich, dass die Peers diese Informationen aktiv an den Indexserver schicken. In diesem Falle genügt es, wenn die Teilnehmer nur diese Einträge an den Indexserver weitergeben, die sich geändert haben, um damit den Traffic zu reduzieren.

Natürlich sind die Indexserver nicht die letzte Instanz und deren Informationen müssen nicht zwingend verbindlich sein. Wenn ein Peer vom Indexserver eine Preisinformation erhält, die nicht mehr aktuell ist, wird der entsprechende Peer, welcher die Datei bereitstellt, mitteilen, dass sich der Preis mittlerweile geändert hat. Der Customer Peer kann dann immer noch entscheiden, ob er den Download mit dem neuen Preis trotzdem durchführen will.

Indexserver sind Schwachstellen im Netzwerk, an welchen das System angegriffen werden kann. Wenn eine Attacke auf diese Stellen ausgeübt wird, können keine Suchanfragen mehr beantwortet werden, und das System bricht zusammen.

Tokenausgabestelle

Da Daten sich einfach replizieren lassen, muss nach erfolgreichem Bezahlungsvorgang sofort sichergestellt werden, dass dieser Peer, welcher das Token abgegeben hat, nicht eine Kopie behält, und damit weiter Dateien kauft. Deshalb muss der Empfänger es von einer vertrauenswürdigen Stelle eintauschen lassen. Diese Aufgabe übernehmen die Tokenausgabestellen. Diese haben die Möglichkeit, Tokens an die Benutzer auszugeben und wieder entgegenzunehmen. Der wichtigste Punkt hier ist die Vertrauenswürdigkeit, damit die Tokens, welche an die Benutzer abgegeben werden, auch ihren Wert behalten. Es ist durchaus möglich, dass sich in einem Peer-to-Peer Netzwerk mehrere Tokenausgabestellen die Arbeit teilen. In diesem Falle wird die Sicherheit noch viel wichtiger: Sobald einer dieser Server gehackt wird und willkürlich Tokens verteilt oder die existierenden Tokens mehrfach an Peers weitergegeben werden, kann das ganze System Schaden nehmen und die Benutzer müssen damit rechnen, dass sie ihr Erspartes verlieren.

Currency Domains

Um dieser Gefahr vorzubeugen, kann zum Beispiel jeder Tokenserver eine Identifikation in seine eigene Tokens codieren, damit man diese Tokens danach eindeutig wieder den Ausgabestellen zuordnen kann [12]. In diesem Falle handelt es sich jeweils um eine andere Currency Domain, so dass die Peers ihre Token jeweils beim richtigen Server eintauschen müssen. Dies hat aber zum Vorteil, dass wenn einmal eine Tokenausgabestelle offline ist, die Peers noch immer mit den Tokens einer anderen Stelle bezahlen können. Wenn eine dieser Stellen jedoch zu viele Tokens in Netzwerk ausgibt, ist es möglich, dass dadurch diese Token an Wert verlieren und nicht mehr eins zu eins gehandelt werden können. Dieser Zustand sollte aber möglichst vermieden werden, weil es dadurch nötig wird, dass jede Datei in der Währung jeder einzelnen Currency Domain bewertet werden muss, was das ganze System dementsprechend komplizierter macht. Eine Vereinfachung davon wäre jedoch, wenn jeder Preis einer Datei der Währung einer Currency Domain zugeordnet werden kann, und dann auch nur mit solchen Tokens bezahlt werden kann. Zwar steigt der bürokratische Aufwand dabei an, weil stets wieder Tokens gewechselt werden müssen, aber die verschiedenen Währungsräume könnten miteinander kommunizieren.

Damit das System nicht schon im kleinen Rahmen zu kompliziert wird, wird in dieser Arbeit auf verschiedene Currency Domains verzichtet.

Supernodes

Sobald das Peer-to-Peer Netzwerk über eine gewisse Grösse anwächst, ist es nicht mehr möglich, dass die Peers mit Hilfe eines Indexservers kommunizieren. Anstatt mehrere weitere Indexserver ins Netz zu stellen, ist es auch möglich, dass dieses Problem mit Hilfe von Supernodes gelöst wird. Supernodes sind normale Peers mit einer besseren Internetanbindung, die zudem auch als Indexserver arbeiten. Damit diese nicht überlastet werden, ist die Zahl der Peers, die sich mit einem Supernode verbinden, um einiges kleiner, als bei echten Indexservern. Dadurch entsteht für den Betreiber des Supernodes nicht so viel Traffic und geringere Kosten. Die Supernodes sind untereinander wiederum verbunden, damit das Netz nicht auseinander fällt. Bei Suchanfragen von Peers können je nach Typ von Peer-to-Peer Netzwerk die Supernodes eine Anfrage an die benachbarten Supernodes weiterleiten oder auch nicht. Wenn dies nicht der Fall ist, werden jedoch die Supernodes den Peers die Adressen von weiteren bekannten Supernodes abgeben, damit die Peers die anderen Supernodes mit der Suchanfrage direkt kontaktieren können.

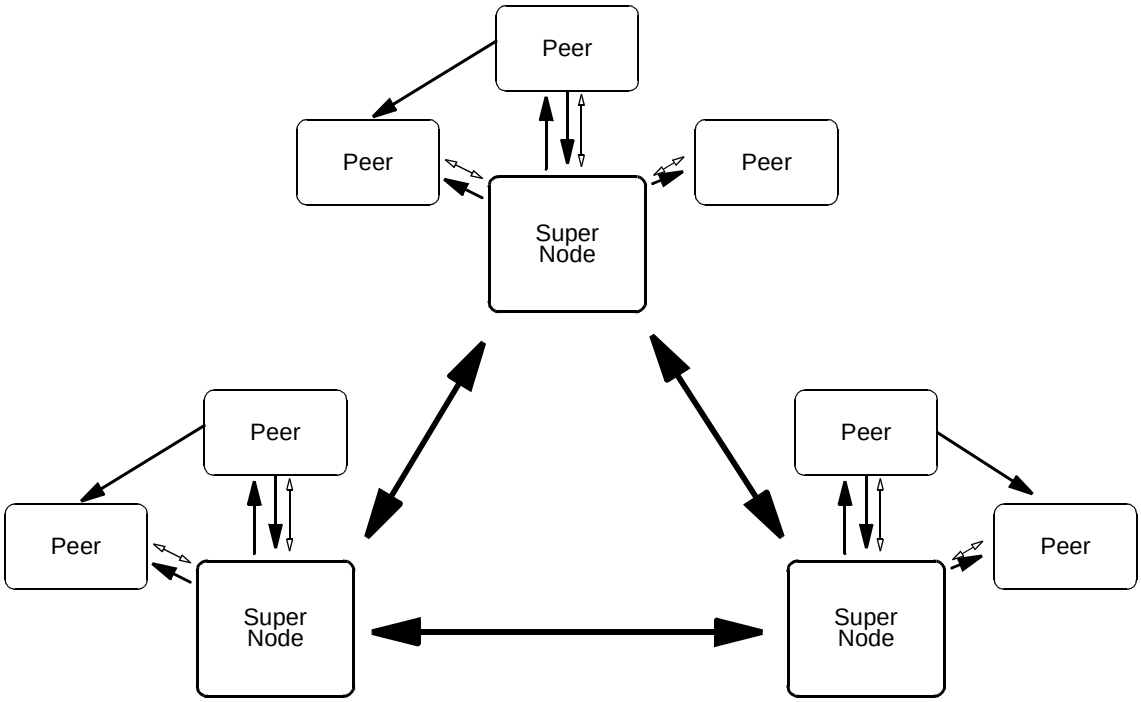

Abb 10: Supernodes agieren als Verbindungsglieder zwischen Subnetzen, damit das Ganze Filesharing Netzwerk besser skaliert.

3.5.3. Skalierbarkeit

Wenn ein Konzept für ein File-sharing System aufgestellt wird, lohnt es sich, auch Gedanken darüber zu machen, wie es sich verhält, wenn sich einmal eine Million oder mehr Peers damit verbinden.

Netze mit einem zentralen Indexerver sind zwar einfach zu implementieren, aber schon bei wenigen tausend Teilnehmern ausgelastet. Wenn stattdessen mehrere zentrale Server genommen werden, kann das Netzwerk noch weiter vergrössert werden, ohne dass es träge wird.

Falls jedoch in Erwägung gezogen wird, dass weltweit Millionen von Benutzer daran teilnehmen sollen, muss die ganze Architektur neu überdacht werden. In diesem Falle macht es Sinn, Super-

nodes einzusetzen, weil diese untereinander schon ein gut vernetztes Peer-to-Peer Netzwerk aufbauen können. Auch wenn diese Supernodes als Indexserver für die einzelnen Peer agieren, bleibt das ganze Netz noch immer ein Peer-to-Peer Netzwerk, da nur die Administration und Suche Client-Servermässig gelöst ist, aber der Austausch von Dateien noch immer direkt zwischen den Peers stattfindet. Sobald einmal mehr als eine Million Peers sich einem solchen Netzwerk angeschlossen haben, ist es nicht mehr so wichtig, dass eine Suchanfrage das ganze Netzwerk durchdringt. In diesem Falle wird nur noch der benachbarte Teil des Netzwerkes durchsucht. Die Wahrscheinlichkeit, dass man darin bereits das findet, wonach man sucht, ist wegen der Durchmischung verhältnismässig gross, da ein grosser Teil des Contents im ganzen Netzwerk auch in diesem Teilast zu finden ist, und der Rest des Netzwerks nicht mehr viel neues bringen würde.

Wenn sich die Peers dann nur in einem Teilast des ganzen Netzwerks aufhalten, kann das Peer-to-Peer Netzwerk beinahe beliebig gross werden, ohne dass die einzelnen Teilnehmer Nachteile in Kauf nehmen müssen. In diesem Falle macht es kaum einen Unterschied, ob es sich nun wirklich um ein grosses Netz oder um mehrere kleine Netze handelt.

3.6. Initialisierung

Sobald das Programm zum ersten Mal mit dem Peer-to-Peer Netzwerk verbunden wird, hat das Programm noch keine Ahnung, unter welchen Adressen andere Peers zu finden sind und inwelchem Rahmen sich die Preise der angebotenen Dateien bewegen. Die einzigen Informationen, die der Client zu Beginn hat, ist die Liste der Dateien, die er selber zum Upload anbietet, und über welche Bandbreite der Computer verfügt. In diesem Falle sei mal vorausgesetzt, dass der Client weiss, mit welchem oder welchen Computern er sich verbinden muss. Nun muss sich das Programm mit den anderen Clients abgleichen, damit es sich in etwa ähnlich verhält wie die anderen. Dies ist vor allem bei der Festlegung der Preise für die angebotenen Dateien wichtig, damit das Programm erfolgreich am Markt teilnehmen kann.

3.6.1. Wie finden sich die einzelnen Peers?

Wir können einmal annehmen, dass dem Client mindestens eine Adresse eines Peers des File-sharing Netzwerks bekannt ist. Diese Adresse kann aus irgendwelchen Quellen stammen. Oft sind bereits einige mögliche IP Adressen in den Konfigurationsdateien des Clients gespeichert. Diese werden dann als erste Anlaufstelle benutzt und stets aktualisiert, während der Client läuft. Beim nächsten Start steht dann wiederum diese aktualisierte Liste zur Verfügung. Wenn das Programm einmal während einer grösseren Zeitspanne nicht ausgeführt wurde, kann es sein, dass keine dieser IP mehr gültig ist. In diesem Falle hat der Client die Möglichkeit, via einer URL eine aktuelle Liste mit Einstiegsknoten zu finden. Bei einigen File-sharing Netzwerken sind diese Adressen so schnelllebig, dass der Benutzer selber in bestimmten Foren nach gültigen Adressen von Servern suchen muss. Sobald einmal ein Peer kontaktiert worden ist, sollte es kein Problem mehr sein, sich mit dem Netzwerk zu verbinden. Die verschiedenen Peers besitzen jeweils eine eigene Liste mit benachbarten Peers oder Supernodes, welche sie untereinander synchronisieren.

Je nach Topographie des Netzwerks sehen diese Verbindungen anders aus: bei kleineren Netzen mit einem oder wenigen Indexservern genügt es für die Clients, einen von diesen zu kennen, um dort die Anfragen zu stellen. Sobald ein Provider Peer für die gesuchte Datei feststeht, wird der Indexserver dessen IP und Port für die direkte Kommunikation an den Peer weiterleiten, der die Suchanfrage gestellt hat. Der Nachteil an der ganzen Sache ist, dass nur die Indexserver das

ganze Netz zusammenhalten. Fallen diese aus, zerbricht das System und die Peers finden sich gegenseitig nicht mehr.

Handelt es sich beim Peer-to-Peer Netzwerk um ein hybrides System mit Supernodes, sieht das Ganze ein bisschen anders aus. Da die meisten Peers die Möglichkeit haben, selber zum Supernode zu avancieren, sind diese nicht so zuverlässig wie andere Indexserver, und es kann durchaus vorkommen, dass diese offline gehen. Jeder dieser Supernodes hat eine Liste mit benachbarten Supernodes, mit welchen er direkt verbunden ist. Diese Liste wird den Peers weitergegeben, welche sich mit dem Supernode verbunden haben. Sobald dieser Node ausfällt, können sie die Liste durchgehen und sich mit dem nächsten möglichen Supernode verbinden. Bei KaZaA [7.4] ist dies durchaus normal, dass der Client von Zeit zu Zeit den Supernode wechselt, damit der Benutzer immer wieder einen anderen Teil des Netzwerks sieht und sich so der Content besser durchmischt. Da die aktuellen Downloads direkt zwischen den Peers stattfinden, hat ein solcher Wechsel keinen Einfluss auf de aktuellen Peer-to-Peer Verbindungen.

3.6.2. Festlegung der Preise

Die Preise im Peer-to-Peer Netzwerk variieren abhängig von Angebot und Nachfrage und können sich mit der Zeit dynamisch ändern. Wenn ein neuer Peer dem Netzwerk beitritt, hat er meist vor, Dateien zum Download anzubieten oder will selber Content beziehen. Er kann aber nicht wissen, wie sich der Markt innerhalb des File-sharing Netzwerks entwickelt hat, seit er das letzte Mal verbunden war. Er kann zwar die Preise, die er bei der letzten Verbindung benutzte, als Anhaltspunkt nehmen, aber es ist nicht sicher, ob diese Werte noch dem Markt entsprechen. Falls der Peer noch nie verbunden war, fehlen ihm sogar diese Informationen. Es bringt nichts, die Preise einmal aufs Geratewohl zu definieren. Je nachdem ob sie viel zu hoch oder niedrig sind, werden sich keine Downloadangebote finden, oder die Uploadbandbreite wird sofort bei suboptimalem Profit voll ausgelastet.

Eine mögliche Lösung wäre, eine Suchanfrage ans Netzwerk zu richten, welche nach den identischen Dateien sucht, welche er selber auch anbietet. Die anderen Peers, welche dieselben Dateien anbieten, werden antworten und den eigenen Preis dieser Dateien durchgeben. Je nach dem, wie aggressiv er im Markt auftreten will wird er den Durchschnitt der Angebote nehmen und diesen ein wenig nach oben oder unten anpassen. Wenn die Preisinformationen der anderen Peers nicht so zuverlässig sind, kann das Programm die höchsten und tiefsten paar Angebote aus der Berechnung entfernen oder aber den Median als Grundlage nehmen.

Für solche Dateien, die nicht in identischer Form im Netzwerk zu finden sind, kann der Client nach Dateien suchen, die in etwa die selbe Grösse haben und der Dateiname zum einem gewissen Grade übereinstimmt. Die Chance ist dort gross, dass es sich um den gleichen Inhalt handelt, wenn auch aus einer anderen Quelle und deshalb leicht in einer anderen Form. Die Preise dieser äquivalenten Dateien können gut als Anhaltspunkt für die Festlegung des eigenen Preises zu Rate gezogen werden. Vielleicht macht es jedoch Sinn, diesen Preis etwas nach unten anzupassen, da diese Datei nur einmal im Netzwerk existiert und die Benutzer demgegenüber eher vorsichtig sind.

Bei Content, für welchen sich keine äquivalenten Dateien finden, ist der Peer für die Festlegung des Preises auf sich gestellt. Wenn der Benutzer in dieser Hinsicht keine Vorgaben gemacht hat, ist es eine gute Idee, dass das Programm anhand der anderen nach obigen Vorgaben festgelegten Preisen ($Price_n$) die Preise für die verbleibenden Dateien proportional zur Dateigrösse (*Size*) errechnet.

$$Preis = \frac{Size}{n} \cdot \sum_{Files} \frac{Price_x}{Size_x}$$

Falls der Benutzer die Preise schon vorgegeben hat, kann es durchaus sein, dass diese nicht mehr den marktüblichen Werten entsprechen, da sich die Preise bereits weiterentwickelt haben. Das kann je nach Situation taktisch Sinn machen, beispielsweise wenn eine Datei als besonders wertvoll angesehen wird, oder wenn der Benutzer den Anschein erwecken will, möglichst viel Content bereitzustellen, auch wenn dieser wegen den überteuerten Preisen wahrscheinlich nie heruntergeladen wird. In jedem Falle sollte das Programm den Benutzer warnen, wenn die Preise signifikant vom Durchschnitt abweichen, so dass der User frühzeitig sieht, dass die Zahl der verdienten Tokens verhältnismässig tief ausfallen wird.

3.6.3. Preisvergleiche mit anderen Peers

Sind einmal die Preise festgelegt, heisst das nicht, dass sich diese nicht mehr ändern können. Je nach Auslastung der Bandbreite der verschiedenen Peers kann sich der Preis einer Datei im Verlaufe eines Tages verändern. Um diesem Umstand Rechnung zu tragen, kann jeder einzelne Peer seine eigenen Preise nach oben oder unten korrigieren, damit die Uploadbandbreite jeweils so stark ausgelastet ist, wie es für den Peer optimal ist. Eine andere Möglichkeit ist, sich an den anderen Peers zu orientieren, welche dieselbe Datei anbieten.

Wenn sich der Peer nur anhand der Downloadofferten orientiert, sieht er zwar, dass zum Beispiel weniger Anfragen eingehen, kann aber nicht genau sagen, ob dies nun nur auf die Streuung zurückzuführen ist, oder ob sich die Preise effektiv geändert haben. Wenn letzteres der Fall ist, muss der Client noch abschätzen können, um wie viel sich die Preise verschoben haben. Die Nachführung der Preise ist daher ein stetiges Approximieren des Optimums.

Falls der Peer die Preise mittels Suchanfrage aktualisiert, weiss er zwar relativ genau, wie teuer diese Datei gehandelt wird, aber dieser Prozess vergrössert den Traffic im Netzwerk nicht unwesentlich. Jeder Peer müsste in diesem Falle in regelmässigen Abständen eine Suchanfrage für jede Datei, die er selber anbietet, abschicken. Dies wäre eine grosse Belastung für die Supernodes, und die Zuverlässigkeit der Informationen ist auch nicht gewährleistet, falls einzelne Nodes falsche Informationen liefern [1]. Eine solche Anfrage kann dann Sinn machen, falls eine bestimmte Datei aus unerklärlichen Gründen über eine längere Zeit nicht angefordert wurde.

3.6.4. Bewertungen und Popularität der Datei

Das Rating und die Popularität haben auch einen Einfluss auf den Preis einer Datei. Da diese Daten idealerweise ein Mittelwert aus möglichst vielen Peers des Netzwerks sind, müssen diese Daten ausgetauscht werden. Ein einzelner Peer kann anhand von eigenen Uploads kaum genau herausfinden, welche Dateien nun beliebt sind, und welche nicht. Falls eine Datei von schlechter Qualität ist, ist es nicht zwingend der Fall, dass die Benutzer der Peers, welche diese Datei anbieten, darüber informiert sind. In diesem Fall könnte ein negatives Rating unter den Peers, welche diese Datei anbieten, weitergegeben werden, damit diese Clients die entsprechenden Benutzer warnen [13], sofern sich solche Ratings häufen, damit dieser Content aus dem Netzwerk entfernt werden kann.

Die Popularität ist ein gutes Indiz, wie häufig eine bestimmte Datei gesucht wird. Die Peers können anhand dieser Information und der Anzahl Peers, welche diese Datei anbieten, ermitteln, wie viel es bringt, dieselbe Datei auch anzubieten. Je grösser das Verhältnis zwischen Nachfrage und Angebot ist, desto mehr Sinn macht es, diese weiter anzubieten, da es in diesem Falle verhältnismässig wenige Quellen hat, um die Nachfrage abzudecken.

$$Nutzen = \frac{Nachfrage}{Angebot}$$

4. Implementierung

Wenn eine Applikation erstellt werden soll, welche die Konzepte des Tokenbasierten Peer-to-Peer Netzwerks umsetzt, genügt es nicht, einfach die Algorithmen zu implementieren. Neben der eigentlichen Aufgabe der Applikation fallen noch viele bürokratisch Aufgaben an, die systemabhängig gelöst werden müssen [11]. Je nach zu Grunde liegender Programmiersprache wird dem Programmierer mehr oder weniger Arbeit abgenommen.

4.1. Programmierumgebung

In diesem Falle wurde für die Implementierung die Programmiersprache Java benutzt, die diese Systemunabhängig ist, und bereits viele Netzwerkfunktionen bietet. Zwar ist Java im Verhältnis zu prozessornäheren Sprachen nicht so schnell, aber die Rechenkapazität ist bei einem File-sharing Client nicht das Bottleneck.

4.1.1. Eclipse

Eclipse ist im Vergleich zu anderen Programmierumgebungen relativ übersichtlich aufgebaut [2]. Man sieht schnell, welche Dateien zum Projekt gehören und welche Funktionen in den verschiedenen Klassen integriert sind.

Während des Programmierens wird in Echtzeit geprüft, ob aufgerufene Funktionen existieren und ob die Datentypen der übergebenen Argumente übereinstimmen. Auch wenn importierte Bibliotheken nicht benötigt werden, wird dies durch Unterstreichen gekennzeichnet.

Die farbliche Trennung von reservierten Wörtern, Strings, Zahlen, Variablen und Kommentar macht den Code übersichtlicher und einfacher lesbar.

Die Entwicklungsumgebung warnt vor fehlenden Strichpunkten, syntaktischen oder einfach zu entdeckenden logischen Fehlern.

Dadurch ist das Programm meist bereits kompilierbar, wenn noch vor dem Kompilieren im Editor keine offensichtlichen Fehler angezeigt werden.

Auf der anderen Seite ist auch Eclipse nicht fähig, bestimmte Laufzeitfehler vor dem Kompilieren zu erkennen, aber das kann von einer Entwicklungsumgebung auch nicht erwartet werden.

4.2. Grundlegende Funktionalität des Programms

Die Applikation hat die Aufgabe, eine Schnittstelle zwischen dem Benutzer und dem Peer-to-Peer Netzwerk zu bilden, so dass der Benutzer nicht zu stark mit den technischen Details des Systems in Berührung kommt, aber trotzdem Einfluss auf das Verhalten des Programms nehmen kann. Ein modularer Aufbau hilft die Übersicht zu behalten, und lässt sich in Zukunft besser erweitern und in andere Projekte einzubauen.

4.2.1. Aufbau des Clients

Der Client ist in verschiedene Klassen aufgeteilt. Dabei wird beim Programmstart in `main` eine Instanz der Hauptklasse `Mainthread` erstellt. Im Prinzip könnte dieser Thread auch statisch sein, aber auf diese Weise kann das Programm leicht so abgeändert werden, dass eine neue Klasse zum Projekt hinzugefügt wird, welche selber ein `main` mitbringt, das das andere im `Mainthread` ersetzt. Diese kann dann beliebig viele Instanzen des `Mainthread` initialisieren, welche dann untereinander kommunizieren können.

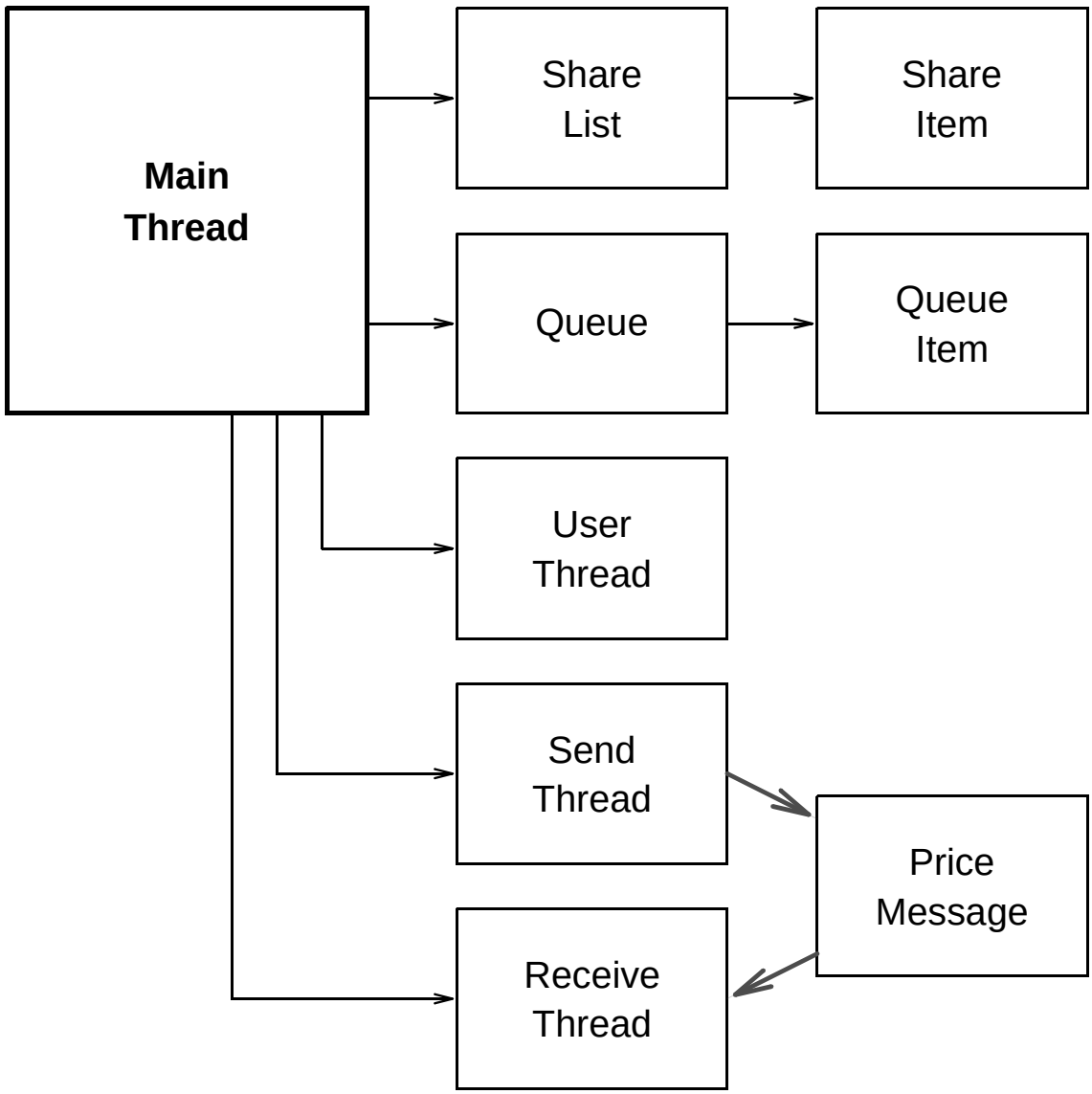

Abb 11: Grundlegender Aufbau der Applikation

In diesem Fall laufen alle diese Clients auf demselben Rechner, so dass sie sich jeweils mit der gleichen IP, aber über verschiedene Ports ansprechen. Wenn die Clients auf verschiedenen Rechnern laufen würden, wäre dann auch die IP unterschiedlich, was aber der Funktionalität keinen Abbruch tut.

Da keine festen Server im Peer-to-Peer Netzwerk existieren, haben die verschiedenen Peers nicht die Möglichkeit, die Adressen von dort zu beziehen. In diesem Falle sind alle Peers direkt untereinander verbunden, wobei davon ausgegangen wird, dass sie jeweils die Adressen voneinander kennen. Bei den Testläufen können die verschiedenen Instanzen die Adressen der anderen Peers von einer Datei einlesen, oder sie sind bereits im Quelltext fest angegeben.

4.2.2. Mainthread

Zu Beginn startet der `Mainthread` eine Instanz des `Receivethread`, damit das Programm von aussen her von anderen Peers her kontaktiert werden kann. Weitere Aufgeben sind das Initialisieren der Preisliste und der Warteschlage für die Downloads, sofern diese nicht leer ist. Die Antworten auf die ausgesendeten Preisanfragen werden vom `Receivethread` empfangen und anhand von dessen Informationen die eigene Liste aktualisiert.

Sind alle Initialisierungen vorgenommen wird eine Schleife gestartet, in welcher in regelmässigen Abständen bürokratische Aufgaben durchgeführt werden, wie das Säubern der Warteschlange und der Listen.

4.2.3. Userthread

Parallel zu den Initialisierungen wird ein `Userthread` gestartet, von dem aus ein Benutzer dem Programm Befehle erteilen kann, wie zum Beispiel das Anpassen von Preisen, das Ausschicken einer Suchanfrage oder der Beginn eines Downloads. Diese Funktion ist in dieser Simulation noch nicht interaktiv, da nur fiktiv Dateien ausgetauscht werden und das Ganze nicht in Echtzeit läuft. Anstelle einer Benutzerschnittstelle bietet aber dieser Thread Platz für das Ausführen eines Skripts oder von Funktionen.

Diese übernehmen die Rolle des Benutzers und geben genau vor, was das Programm zu tun hat. Andererseits ist auch möglich, dass dem Programm nur ein Ziel gesetzt wird, das ein Algorithmus in diesem Thread mehr oder weniger gezielt zu erreichen versucht.

Da in der Simulation nicht echte Daten hin und her geschoben werden, ist es schwierig zu bewerkstelligen, dass die Simulation auf allen Peers mit der gleichen geschwindigkeit läuft. Um die Unterschiede zwischen kleinen und grossen Dateien zu verdeutlichen, wird nach dem simulierten Sendeprozess einer virtuellen Datei ein bestimmter Zeitraum gewartet, bevor das Programm weiterarbeitet. Dies wird mit der Methode `Thread.sleep()` erreicht, wobei die Wartedauer in Millisekunden proportional zur Dateigrösse anwächst. Da in diesem Falle nur ein Thread schläft, werden die anderen Funktionen wie das Empfangen von Paketen nicht beeinträchtigt.

4.2.4. Receivethread

Dieser Thread ist, wie aus dem Name geschlossen werden kann, für den Empfang von Mitteilungen zuständig. Da diese Messages zu einem beliebigen Zeitpunkt eintreffen können, muss dieser Thread stets aktiv sein und am entsprechenden Port horchen. Um dies zu erreichen, wird beim Programmstart eine Instanz dieser Klasse im Hintergrund gestartet, damit diese auf die verschiedenen Ereignisse warten kann und dabei die anderen Threads nicht blockiert.

Der Thread besteht hauptsächlich aus einer Endlosschleife mit einer Abbruchbedingung. Innerhalb dieser Schleife wird eine Funktion von Java aufgerufen, welche auf Pakete an einem bestimmten Port wartet. Diese Funktion ist aber blockierend, so dass die Ausführung des Threads jeweils an dieser Stelle angehalten wird, wenn gerade keine Pakete eintreffen.

Sobald ein Paket ankommt, wird es so schnell wie möglich analysiert und an die entsprechenden Programmteile weitergegeben, welche sich dann je nach Typ des Pakets um die Antwort oder die Aktualisierungen in der Datenbank kümmern.

Da dieser Thread stets auf neue Pakete wartet, ist es nicht möglich, diesen mit Hilfe einer Abbruchbedingung direkt zu beenden, wenn das Programm beendet werden soll. Der Thread kann diese Abbruchbedingung nur überprüfen, nachdem ein Paket eingetroffen ist und die blockierende Funktion kurzzeitig verlassen wird. Auch das Killen eines Threads am Ende des Programms ist verpönt. Um das Programm dennoch zu beenden, setzt der `Mainthread` oder `Userthread` eine globale boolesche Variable, welche signalisiert, dass das Programm beendet werden soll. Danach schickt sich das Programm selber ein Paket, welche dem Receivethread ermöglicht, aus der Blockierung der Java-Funktion zu entkommen, die Abbruchbedingungen zu überprüfen und den Thread sauber zu beenden.

4.2.5. Sendthread

Analog zum Empfangsthread, ist dies die Klasse, über welche die Pakete das Programm verlassen. Alle Methoden des Clients, welche Pakete an andere Peers schicken wollen, rufen für diesen Zweck den Sendthread auf. Dieser Thread ist nicht blockierend; wenn mehrere Programmteile gleichzeitig Daten zu senden versuchen, werden zwei Threads generiert, die dann parallel ihre Daten versenden.

Ist das Paket einmal abgesendet, beendet sich der Thread selbständig, da für das nächste Paket ein neuer Thread generiert wird. Diese Messages sind also nicht verbindungsorientiert, damit kein Programmteil hängen bleibt, wenn einmal ein anderer Peer überraschend nicht mehr auf die Anfrage antwortet.

4.2.6. Sharelist

Diese Klasse verwaltet die angebotenen Dateien. Sie erstellt eine Instanz von der Klasse `Shareitem` für jede Datei, die im System für den Tausch freigegeben wurde. `Shareitem` enthält alle wichtigen Informationen zu einer bestimmten Datei, wie dessen Name, Eigenschaften, aber auch Preis oder Beliebtheit.

Die Methoden von `Sharelist` übernehmen alle wichtigen Funktionen, welche mit dem Handling der Dateien zu tun haben, wie zum Beispiel das Auslesen der Eigenschaften, das Anpassen der Preise und deren Beliebtheit. Wenn der Client einmal ausgebaut wird, wird diese Klasse auch die entsprechenden Dateien lesen und die angeforderten Fragmente an den `Sendthread` überreichen.

4.2.7. Queue

Analog zu `Sharelist` ist die Klasse `Queue` für die Downloadofferten zuständig. Sobald andere Peers anbieten, eine Datei zu einem bestimmten Preis herunterzuladen, wird diese Offerte an diese Klasse weitergereicht. `Queue` erstellt für jede Downloadofferte eine Instanz der Klasse `Queueitem`, welche jeweils die Informationen zu der jeweiligen Offerte, wie Preis, Verfallsdatum oder die Adresse des anfragenden Peers speichert. Die Anzahl Offerten ist sowohl gesamthaft, wie auch pro Benutzer beschränkt, damit das System gegen Angriffe besser geschützt ist.

`Queue` geht regelmässig die Liste durch und entfernt die abgelaufenen und ungültigen Einträge. Sobald der `Sendthread` nicht ausgelastet ist, fragt diese nach der besten Offerte in der `Queue` und offeriert diesem Peer ein Download.

4.2.8. Pricemessage

Um einfacher herauszufinden, um was für eine Message es sich bei einem eintreffenden Paket handelt, wird bei der ganzen Kommunikation nur ein Objekt versendet. Dieses für diesen Zweck erstellte Objekt wird hier `Pricemessage` genannt, und enthält neben einer Kennzeichnung des Pakettyps Felder für sämtliche benötigte Informationen.

Je nach Pakettyp werden einige Felder von `Pricemessage` nicht benötigt und bleiben leer. Bei Integervariablen ist zwar dieser Speicherplatz verloren, dafür aber nicht so gross. Falls die Strings nicht benutzt werden, wird einfach ein leerer String gesendet, für welchen in diesem Falle nur wenige Bytes reserviert werden und daher das Paket nicht aufblähen. Dasselbe gilt für das Datenfeld, fall keine Dateifragmente gesendet werden.

Da es sich bei jedem eintreffenden Paket um ein Objekt des Typs `Pricemessage` handelt, kann der `Receivethread` einfach die Kennung auslesen und anhand dieser das Paket dem richtigen Thread weiterleiten.

4.3. Parallelität

Die parallele Ausführung verschiedener Threads ist eine wichtige Eigenschaft des Clients. Dies ist vor allem der Fall, weil das Programm nicht wissen kann, wann eine Anfrage von aussen eintrifft. Diese müssen nämlich entgegengenommen werden, während gleichzeitig Daten gesendet werden oder der Benutzer selber mit dem Programm interagiert. Dadurch kann es zudem vorkommen, dass sich diese Threads gegenseitig aufrufen. In diesem Falle darf weder ein Deadlock noch eine Endlosschleife entstehen. Nehmen wir mal folgendes Szenario: Ein Peer sendet uns eine Anfrage für einen Download, welche wir akzeptieren und dafür ein Token anfordern. Bevor nun der Client beginnt, Teile der Datei zu senden, wartet dieser auf das Token. Wenn dieses eintrifft, muss es erst auf die Authentizität geprüft und eingetauscht werden. Für das Eintauschen ist aber auch der Thread zuständig, welcher das Senden übernimmt. Dieser Thread ist aber noch damit beschäftigt, Teile der Datei zu senden, was aber momentan auf Eis liegt, da die Echtheit des Tokens noch nicht bestätigt ist. In diesem Falle muss ein Deadlock verhindert werden, indem man die Funktionalität der Threads, welche für das Senden und für das Empfangen von Paketen zuständig sind, auf genau diese Funktionen einschränkt und für jeden Sendeprozess parallel einen neuen Sendthread startet.

4.3.1. Virtuelle Parallelität

Obengenanntes Problem wird dadurch gelöst, indem zum einen keine festen Verbindungen aufgebaut werden und zum anderen für jede Aufgabe ein eigener Thread gestartet wird.

Wenn also ein Peer ein Fragment einer Datei anfordert, aber noch kein Token eingetroffen ist, wird in diesem Falle die entsprechende Methode im Sendthread kein Fragment der Datei senden, sondern nur die Aufforderung, ein Token zuzuschicken. Nachdem diese Aufforderung abgeschickt wurde, wird dieser Thread beendet. Trifft nun ein Token ein, wird dieses dem entsprechenden Peer gutgeschrieben. Sobald dies passiert ist, kann dieser Peer nun erneut versuchen, das gewünschte Fragment zu bestellen.

In der Zwischenzeit ist der Client permanent offen für neue Anfragen und wird diese Parallel dazu beantworten. Das kann aber auch Nachteile haben: falls nämlich in der Anfrage nach einem Token und dem Senden des ersten Dateifragments ein anderer Peer seinerseits ein Fragment anfordert, für welches er schon bezahlt hat. So ist es durchaus möglich, dass der erste Thread ein Token sendet, aber dann mangels Uploadbandbreite warten muss. In diesem Falle kommt der Download nicht sofort zu Stande, und der Customer Peer hat die Möglichkeit zu warten oder aber einToken zurückzuverlangen.

5. Experimentelles Vergleichen der Preise

Dieses Experiment soll zeigen, wie sich die Preise von angebotenen Dateien verändern, während die Peers im Filesharing Netzwerk Preisinformationen austauschen. In diesem Fall schicken die Peers ihre Preisinformationen nicht an alle benachbarten Peers, sondern werden von denjenigen Peers angefragt, welche eine Preisinformation benötigen (Pull-Methode). Die empfangenen Preise werden gewichtet und einen entsprechenden Mittelwert mit dem bisherigen Preis errechnet. Falls die Preisantwort zu weit von dem bisherigen Wert entfernt ist, interpretiert das Programm diese Message als Falschmeldung und ignoriert sie. Aus diesem Grunde ist es nicht möglich, dass sich die Preise innerhalb einer kurzen Zeit sprunghaft ändern. Alternativ dazu könnten ungewöhnlich stark differierende Preise einfach weniger gewichtet werden, damit bei einem Preiszerfall die Preise lokalen Preise nicht auf einem hohen Niveau hängen bleiben.

Die Simulation wurde so implementiert, dass beim Programmstart eine grosse Zahl Threads generiert wird, die jeweils auf einem anderen Port horchen. Die Anzahl der Peers kann als Argument dem Programm übergeben werden, so wie der Port, welcher dem ersten Thread zugeordnet ist. Der zweite Thread benutzt dann einfach den nächst höheren Port und so weiter. Auf diese Weise wird sichergestellt, dass die einzelnen Threads a priori die Ports der anderen Threads kennen.

Die Preise der vordefinierten virtuellen Dateien, welche die Peers zur Verfügung stellen, unterscheiden sich untereinander zufällig in einem gewissen Rahmen. Das Ziel der Peers ist es nun, diese Preise untereinander abzugleichen. Dabei benutzen sie keine speziellen Taktiken, sondern erkundigen sich lediglich bei den benachbarten Peers nach deren Preisen. Diese Preisantworten fliessen dann in die eigenen Preise ein.

Zurzeit ist die Gewichtung dieser Preise noch konstant, aber es ist auch möglich, diese Gewichtung abhängig von der Reputation des benachbarten Peers zu variieren.

6. Ergebnisse

Wie erwartet gleichen sich die Preise gegenseitig an. Dieser Prozess verläuft stetig, aber nicht vollständig. Da die Preise diskret sind, bleiben kleine Unterschiede stets bestehen, weil bei der Gewichtung diese feinen Unterschiede herausgerundet werden.

Auch ist es nicht zwingend der Fall, dass sich die Preise beim Mittelwert aller Peers ansiedeln. Je nach dem, welche Peers am Anfang eher häufiger gefragt wurden, kann sich das Gleichgewicht in diese Richtung verschieben.

Natürlich muss noch angemerkt werden, dass sich die Peers unvoreingenommen verhalten; die Popularität und Verbreitung einer Datei hat in dieser Rechnung noch keinen Einfluss auf die Preise.

Je grösser die Zahl der Peers, desto länger dauert dementsprechend der Angleichungsprozess. Dabei muss aber beachtet werden, dass wenn alle diese Peers von einem Computer simuliert werden, die Geschwindigkeit der Simulation einbricht, da dieser einzelne Computer nur eine virtuelle Parallelität simulieren kann, aber im Hintergrund alle Peers nacheinander abarbeitet. In einem grösseren Verbund ist so die Effizienz wesentlich grösser, da die einzelnen Datenströme an sich ausbalanciert sind, und einander nicht in die Quere kommen. Trotzdem haben in grösseren Netzwerken die Peers nicht mehr die Möglichkeit, alle anderen Anbieter zu kontaktieren. Der Mittelwert rechnet sich in diesem Falle aus einer zufälligen Auswahl von Peers. Der Fehler ist statistisch gesehen klein, und falls es trotzdem einige ungewollte Abweichungen gibt, existieren auf dem Rechner weitere Dateien, dessen Preise statistisch unabhängig eruiert wurde, so dass dann einfach diese als Ersatz bevorzugt getauscht werden. Durch diese zufällige statistische Streuung wird zudem eine weitere Verteilung der Downloadanfragen erreicht.

7. Verwandte Systeme

7.1. Napster

Napster ist der Pionier unter den File-sharing Tools, welche auf einem Peer-to-Peer Netzwerken aufbauen. Es war von einer einzelnen Person entwickelt worden und war auf den Austausch vom mp3 Dateien spezialisiert. Zu diesem Zeitpunkt gab es praktisch nur Serversysteme, von welchen man Dateien beziehen konnte, sofern man die entsprechenden Zugangsdaten hatte. Da diese Server meist nur eine kleine Auswahl an Dateien und eine kurze Lebensdauer hatten, machte es Napster den Benutzer wesentlich einfacher und gewann in kurzer Zeit stark an Popularität. Auch wenn Napster von vielen Leuten als schlecht für die Wirtschaft angesehen wurde, kann man nicht verleugnen, dass dieses File-Sharing Netzwerk viel zur Verbreitung von Breitbandanschlüssen und in diesem Sinne auch zur Vergrösserung der Kapazitäten im Internet beitrug.

Einzelne zentrale Server, auf welche sich die Benutzer des Netzwerkes anmelden, waren für die Suchanfragen zuständig. Falls eine Anfrage erfolgreich war, initiierten sie den Dateiaustausch, welcher dann direkt zwischen den Peers erfolgte. Sobald diese Server ausfielen, brach auch das ganze File-Sharing Netzwerk zusammen. Diese Server wurden dann auch zum Angriffspunkt, als die Musikindustrie Klagen an Napster richtete. Die Betreiber konnten sich nicht mehr gegen diese Macht wehren, und so wurde Napster deaktiviert, was aber vielen Nachahmern die Möglichkeit gab, ihre Entwicklungen auf den Markt zu bringen.

7.2. Gnutella

Gnutella wurde zu der Zeit, als Napster weltweit populär war, als Open Source von Justin Frankel (Urheber von Winamp) implementiert und von der Gnu Community weiterentwickelt. Das Ziel von Gnutella war es, ohne zentralen Server auszukommen, so dass es nicht möglich ist, das Netzwerk mit einer Attacke auf wenige Peers lahmzulegen, oder mit Hilfe rechtlicher Mitteln zu schliessen, wie es bei Napster passiert war.

Sobald ein Peer die Verbindung zum Gnutella Filesharing System aufgebaut hat, ist dieser ein gleichwertiges Mitglied, wie die anderen Peers. Suchanfragen werden per Broadcast weitergegeben; die Empfänger antworten mit ihren ergebnissen direkt an den Peer zurück. Da der Quellcode frei benutzbar ist, kamen innerhalb von kurzer Zeit verschiedene Implementationen auf den Markt.

Da bei einer wachsenden Zahl von Peers der durch die Anfragen verursachte Traffic dementsprechend zunimmt, Skaliert das System nicht so gut. Dies wurde vor allem dann deutlich, als Napster offline ging und viele Benutzer zu Gnutella wechselten. Die Performance des Systems brach innerhalb von kurzer Zeit ein.

Um dem vorzubeugen, wurde am Protokoll weiterentwickelt, so dass daraus ein Gnutella2 entstand.

7.3. Mojo Nation

Mojo Nation ist ein Versuch, die Loadbalancierung mit Hilfe von einer virtuellen Währung zu implementieren [10]. Die entsprechende Währungseinheit heisst, wie der Name des Netzwerks schon andeuten lässt, Mojo. Für den Download von jeder Datei muss der Interessent eine bestimmte Anzahl Mojos bezahlen. Der Preis kann vom Provider Peer bestimmt werden, wobei es auch möglich ist, dass er den Content gratis abgibt, also keine Mojos dafür verlangt.

Eine weitere spezielle Eigenschaft von Mojo Nation ist, dass die Dateien jeweils redundant in 8 Stücke aufgeteilt werden, so dass die ursprüngliche Datei nach dem Download von beliebigen 4 Teilen wieder hergestellt werden kann. Der Sinn dahinter ist, dass wenn ein Teil der 8 Stücke im Mojo-Netzwerk gerade nicht erhältlich ist, der Download trotzdem noch abgeschlossen werden kann. Andererseits ist die Datei in der zerstückelten Form für jemand, der sie auf dem lokalen Computer anbietet zu dieser Zeit nicht brauchbar, da sie von den anderen Anwendungen nicht lesbar ist.

Nach einer gewissen Zeit wurde jedoch das Projekt abgebrochen, noch ehe es im Internet Fuss fassen konnte. Die Ursachen, warum hier nicht weiterentwickelt wurde, sind mir nicht bekannt.

7.4. KaZaA

KaZaA ist einer der ersten Filesharing Tools, welcher nach der Schliessung des Pioniers Napster dessen Marktnische übernommen hat. Es basiert auf der FastTrack Technologie, welche von Sherman Networks vertrieben wird. Das Netzwerk wird von den Clients KaZaA, Grokster und Morpheus benutzt, wobei letzterer vor einiger Zeit wegen wirtschaftlichen Gründen aus dem Verbund ausgeschlossen wurde und zu Gnutella wechseln musste. Auch wenn der Client gratis erhältlich ist, ist das ganze Tool trotzdem kommerziell und der Quellcode nicht public. Sherman Networks verlegte seinen Geschäftssitz nach Vanuatu, um nicht mit amerikanischen und europäischen Gesetzen in Konflikt zu kommen.

7.4.1. Dateisystem

Damit das System gut skaliert, haben Nodes mit einer guten Internetanbindung die Möglichkeit, als Supernode im System teilzunehmen, welche dann von den verschiedenen Customer Peers kontaktiert werden. Jeder Supernode ist zudem mit 200 weiteren Supernodes verbunden, welche so einen zufälligen Graphen aufspannen, welcher möglichst vielfach zusammenhängend ist. In der Regel sind ein grosser Teil dieser Supernodes aus der gleichen Region (USA, Europa, Asien); es existieren aber auch eine gewisse Anzahl globale Verbindungen, damit der kürzeste Weg von einem zu einem anderen Node nicht zu lang ist.

Wird von einem Knoten im Netz eine Suchanfrage ausgegeben, so wird der Supernode, an dem man hängt, diese Anfrage an alle benachbarten Supernodes weiterleiten. Diese beantworten dann die Suchanfrage, aber leiten diese nicht mehr weiter. Man sieht als Benutzer also nicht das ganze Netzwerk, sondern nur den Teil davon, welcher maximal zwei Spernodes entfernt ist. Zudem verbindet sich der Client sporadisch wieder mit einem neuen Supernode, so dass man nicht immer nur im gleichen Teil des Peer-to-Peer Netzwerks sucht. Mit neueren Versionen des Clients ist es für den Benutzer zudem möglich, aktiv in die Auswahl der Supernodes einzugreifen, da der Content zu einem gewissen Grade auch von der Region abhängt.

Sobald die gesuchte Datei gefunden wird, kann der Download beginnen. Von Vorteil ist auch, wenn dieselbe Datei von mehreren Peers bezogen werden kann, damit man nicht von einer Quelle abhängig ist. Ein Hashcode stellt sicher, dass es sich effektiv um dieselbe Datei handelt,

auch wenn der Dateiname variiert. Der Client ist zudem fähig, während des Downloas nach weiteren Quellen zu suchen, und von denen gleichzeitig Fragmente der Datei anzufordern, was den Durchsatz optimiert. Die IP-Adressen der verschiedenen Quellen einer Datei werden zudem lokal gespeichert, damit nicht bei jedem Neustart die Suche wieder von vorne beginnen muss. Natürlich gibt es darunter immer wieder Quellen, welche von der Bildfläche verschwinden, die IP ändern oder das gesuchte File nicht mehr anbieten. In diesem Falle werden diese nach einer bestimmten Zeit gelöscht, sofern noch genügend andere Quellen vorhanden sind; falls jedoch die Zahl der Peers eher klein ist, kommt es nicht so darauf an, und die Adressen werden behalten, für den Fall, dass diese Quelle ins Leben zurückkehrt.

Wird neuer Content ins Netzwerk geladen, so ist dieser nicht sofort im ganzen Netz zu finden. So werden zuerst in der näheren Umgebung wenige Leute diese Datei runterladen und selber wieder zur Verfügung stellen. Da die Peers und Supernodes sich immer wieder neu untereinander anordnen, verbreitet sich die Datei immer mehr im ganzen Netzwerk, sofern sie in allen Regionen gleich beliebt ist. Bei lokalem Content muss man aber davon ausgehen, dass dieser nicht überall im Netz zu finden ist, sondern nur in einem bestimmten Gebiet.

7.4.2. Lastbalancierung

Um die einzelnen Benutzer zu ermuntern, selber Dateien bereitzustellen, wurde ab der zweiten Version ein Participation Rating eingeführt, welches angibt, wieviel Content man selber ins Peer-to-Peer Netz hinauflädt. Dabei ist nicht entscheidend, wie viel man bereitstellt, sondern wie viel effektiv von anderen Benutzern geladen wird. Diejenigen Benutzer mit einer besseren Bewertung haben bei anderen Peers, bei welchen die Zahl der Downloadslots beschränkt ist, eine höhere Priorität, einmal an die Reihe zu kommen. So wird im Durchschnitt eine Datei in kürzerer Zeit heruntergeladen, als bei Benutzern mit einer niedrigen Bewertung.

Dieses System ist aber trotz nicht veröffentlichtem Quellcode nach kurzer Zeit geknackt worden, so dass die Benutzer ihr Rating immer aufs Maximum setzen können, und so das System wieder ähnliche Eigenschaften hat, wie vorher. Zurzeit nutzen etwas weniger als die Hälfte diesen Cheat. Am meisten bringt dies beim Tauschen von Videodateien, weil dort die Nachfrage wesentlich höher ist, als das Angebot. Zudem ist neben dem Speicherplatzbedarf dort auch die psychologische Hemmschwelle, etwas anzubieten, höher. Bei Audiodateien scheint sich das ganze System so weit ausnivelliert zu haben, dass die meisten Leute schon besitzen, was sie suchen, so dass die Provider Peers die Nachfrage schnell bewältigen können.

7.4.3. Bewertung des Contents

Da in letzter Zeit immer mehr Dateien, deren Name nicht mit dem Inhalt übereinstimmt, und solche von schlechter Qualität auftauchen, wurde eine Möglichkeit eingeführt, diese zu bewerten. In einem Peer-to-Peer Netzwerk ist es aber nur schwer möglich, alle Bewertungen zentral zu speichern, und so wird die Bewertung eines Files jeweils von einem Benutzer lokal mit der Datei abgelegt. Wie viel versprechend der Ansatz auch klingen mag, hat doch dieses System einige Nachteile. Wenn zum Beispiel ein Benutzer eine grosse unbrauchbare Datei ins Netz stellt, und diese selber als gut bewertet, werden sich Opfer finden, welche diese Datei herunterladen. Sobald dies geschehen ist, werden diese vielleicht merken, dass diese Datei unbrauchbar ist. Damit aber die anderen Benutzer davon erfahren, muss dieser User dieser Datei schlecht bewerten und den anderen wieder zum Download zur Verfügung stellen. Falls er dies nicht macht, sondern die Datei gleich löscht, ist seine Bewertung nicht sichtbar, sondern wird gleichzeitig mit der Datei gelöscht. Es müssen sich also immer Leute zur Verfügung stellen, die solche Fakes als solche bezeichnen und im Netzwerk zur Verfügung stellen, auch wenn davon ein guter Teil des lokalen Speicherplatzes benutzt wird.

Es gibt aber auch Peers im Netzwerk, welche nur viel Content ansammeln und dann wieder zur

Verfügung stellen, da sie keine Zeit haben, die Dateien zu begutachten und zu bewerten. In diesem Falle kann über die Qualität der Datei nichts augesagt werden, und man muss hoffen, dass sich ein anderer Provider Peer dazu äussert.

Das Bewerten der Dateien macht dann Sinn, falls eine Datei von guter Qualität ist. In diesem Falle ist die Wahrscheinlichkeit grösser, dass der Peer die Datei auf dem Computer lässt, gut bewertet und wieder anbietet. Zwar kann es auch hier vorkommen, dass einzelne Benutzer eine falsche Bewertung abgeben, aber der Mehrheit kann man schon vertrauen.

Parallel zu den Bewertungen besteht aber immer noch die Möglichkeit, die Benutzer, von welchen man eine Datei bezieht, direkt via den eingebauten Messenger zu kontaktieren und anzufragen. Wenn diese Benutzer am Computer sind. ist die Chance nicht all zu schlecht, dass man sogar eine Antwort erhält. Dies ist aber nicht sö häufig der Fall, da viele Clients im hintergraund 24 Stunden am Tag laufen. Zudem kommt es sogar vor, dass man vom Provider Peer gewarnt wird, wenn man einen Fake zu downloaden beginnt. Irgendwo befinden sich hier noch immer Menschen hinter dem System.

7.5. eDonkey

Im Gegensatz zu KaZaA (Fasttrack Netzwerk) ist eDonkey Opensource, so dass mittlerweile verschiedene Clients erschienen sind, die das Netzwerk nutzen. Weiter sind einige Implementierungen so konstruiert, dass sie sich mit verschiedenen Peer-to-Peer Netzwerken verbinden können, so dass auch zwischen den verschiedenen Peer-to-Peer Netzwerken eine Durchmischung der Dateien stattfindet. Eine sehr gute Beschreibung dazu findet sich in der Hilfedatei des eMule Clients, welcher dasselbe Protokoll nutzt [6].

Das eDonkey Netzwerk hat sich auf den Austausch von grossen Dateien spezialisiert (in der Regel mehr als 50 Megabyte), wobei diese jeweils in Blöcke zu 9 Megabytes aufgeteilt werden. Diese Aufteilung geschieht aber nur virtuell; auf der Festplatte ist die Datei als Ganzes zu sehen, auch wenn Teile davon noch unvollständig sind. Jede dieser Blöcke erhält einen eigenen Hashwert, so dass diese eindeutig als Teil des Originalfiles erkannt werden können.

7.5.1. Austausch von Fragmenten unvollständiger Dateien

Sehr oft kommt es vor, dass Benutzer sich nur in ein Netzwerk einwählen, um eine bestimmte Datei herunterzuladen. Sobald dies geschehen ist, brechen diese Benutzer die Verbindung ab oder nehmen die Datei aus dem Sharedfolder heraus und stellen sie nicht weiter der Community zur Verfügung. Durch die Aufteilung einer Datei in die verschiedenen gehashten Blöcke wird es möglich, diese bereits wieder dem Netzwerk zur Verfügung zu stellen, bevor der Download einer ganzen Datei abgeschlossen ist. Während nämlich die Clients Teile einer Datei herunterladen, geben sie meist abgeschlossene Fragmente derselben Datei zum Upload frei. Algorithmen im System sorgen dafür, dass die Blöcke einer Datei nicht sequentiell von vorne nach hinten angefordert werden, sondern dass die Reihenfolge zufällig gewählt wird. Dadurch wird gewährleistet, dass nicht alle Benutzer jeweils die gleichen Teile bereits besitzen, sondern dass die Häufigkeit eines jeden Fragments einer Datei bei den verschiedenen Benutzern etwa gleich hoch ist. So ist es nicht nötig, dass jemand die Datei vollkommen besitzt um sie zum Download freizugeben, sondern auch die Benutzer mit einigen Fragmenten können untereinander Fragmente austauschen und so gegenseitig die ganze Datei vervollständigen.

Damit das System gut skaliert, wird darauf geachtet, dass die einzelnen Benutzer nicht zu viele kleine Dateien zur Verfügung stellen, sondern wenige grosse. Bei Audiodateien werden so zum Beispiel ganze Alben in Archive zusammengefasst. Dadurch ist die zur Verfügung gestellte

Menge an Daten noch immer hoch, aber die Zahl der Dateien ist überschaubarer, und es geht weniger Bandbreite verloren, wenn die Customer dem Server die eigene Liste mit freigegebenen Daten übermitteln oder Suchen durchgeführt werden.

7.5.2. Netzstruktur

Ähnlich wie bei KaZaA basiert das eDonkey Netzwerk auf einer hybriden Netzwerkstruktur. Die einzelnen Clients verbinden sich zu einem Server, welcher die Contentliste der Verbundenen Clients nachführt und die Suchanfragen bearbeitet. Sobald die Ergebnisse zurückgesendet worden sind, verbinden sich die Clients selber untereinander. Für diesen Zweck braucht aber jeder Client eine IP mit freiem TCP Port 4662 und UDP Port 4672. Ist dies nicht möglich, weil der Client sich in einem lokalen Netzwerk befindet, das nach aussen nur eine IP besitzt, kriegt er nur eine niedrige Identifikationsnummer und wird so zu einem Teilnehmer zweiter Klasse. Diese können untereinander nicht kommunizieren, sondern müssen alles über einen Server leiten. Dies belastet den Server, was zur Folge haben kann, dass einzelne Server die Zahl der Clients mit niedriger ID beschränken oder diese ganz abweisen.

Sind diese Ports jedoch erreichbar, können diese Teilnehmer direkt untereinander Informationen austauschen und belasten so den Server nicht. Dadurch ist die Zahl der Clients, die mit einem einzelnen Server verbunden sind, höher als bei KaZaA.

Die Server sind untereinander schwach verbunden, was in diesem Falle heisst, dass sie jeweils eine Liste von weiteren Servern besitzen, welche sie jeweils den Clients weitergeben.

Im Gegensatz zu KaZaA leitet der Server eine Suchanfrage nicht weiter, sondern beantwortet sie selber mit Hilfe der Dateiliste, welche die Informationen der direkt mit ihm verbundenen Clients enthält. Um eine umfassendere Suche zu ermöglichen, muss der Client selber die verschiedenen Server anfragen, welche sich in seiner Liste befinden.

Sobald ein Server in dieser Liste eine bestimmte Zahl von Anfragen nicht beantwortet, wird dieser aus der Liste gelöscht. Zudem wird diese Serverliste bei Verbindungsaufnahme mit einem Server oder anderen Client untereinander aktualisiert.

7.5.3. Kreditsystem

eDonkey besitzt ein internes Kreditsystem, welches Benutzer belohnt, die selber aktiv einen Beitrag an die Community leisten. Da das ganze System open Source ist und keine zentrale Stelle besitzt, kann auch das Krediting nicht zentral geregelt werden. Aus diesem Grund vergeben die einzelnen Clients untereinander gewisse Credits. Um einer Manipulation vorzubeugen, verwaltet nicht der Besitzer der Credits seine eigenen Punkte, sondern jeweils die Gegenstelle, bei welcher er die Punkte einlösen kann. Dieser Credit ist ein Wert zwischen 1 und 10, welcher aus dem Verhältnis von Upload und download errechnet wird.

Es macht Sinn, dass jeder Teilnehmer mittels eines Hashes, welcher bei der ersten Ausführung des Clients generiert wird, wieder erkannt wird. Diese Credits kommen dann zum Einsatz, wenn unter den Quellen einer gesuchten Datei sich ein Peer befindet, welchem man selber schon Content heraufgeladen hat. Je mehr die Anzahl der Benutzer des Netzwerks ansteigt, desto geringer wird die Wahrscheinlichkeit, dass dies vorkommt, da man kaum eine Verbindung mit hunderttausenden von Rechnern gehabt haben kann. Ein Dateiaustausch ist also auch dann möglich, wenn man sein Gegenüber noch nicht kennt.

Sobald eine Datei gefunden ist, welche der Customer Peer herunterladen will, kommt er auf eine Warteliste. Die Position in dieser Queue ergibt sich aus dem Produkt aus der eigenen Bewertung und Wartezeit in der Schlange/100 Sekunden. Die eigene Bewertung errechnet sich aus einem

Startwert von hundert Punkten, multipliziert mit den eigenen Credits und anderen Faktoren wie Popularität der Datei.

Mit wachsender Wartezeit schreitet man in der Warteschlange immer weiter nach vorne, bis der Download beginnen kann. Weiter ist es möglich, als Benutzer Freunde zu definieren, welche bevorzugt behandelt werden.

8. Zusammenfassung

Durch die Einführung der Tokens als Währung wird sich die Last etwas ausbalancieren. Es wird nicht mehr möglich sein, sich ausschliesslich von anderen Peers zu bedienen, ohne selber etwas an das Netzwerk zurückzugeben. Ob das tokenbasierte Filesharing Netzwerk eine Chance gegen die anderen Tauschbörsen hat, wird sich noch zeigen. Auf der einen Seite werden durch die Bedingung, dass man selber Content zur Community beitragen muss, viele Freeloader zu anderen Tauschbörsen wechseln. Dies hat aber insofern keinen Nachteil, als diese sowieso keine Dateien zum System Beitragen und deshalb nur das Peer-to-Peer Netzwerk belasten würden. Es wird auch in diesem Netzwerk altruistische Peers geben, welchen ihren Content gratis zur Verfügung stellen. Diese müssen aber nur als kleine Zugabe angesehen werden, da sie dementsprechend begehrt sind, dass ein Download nur aufwändig zu ergattern ist. Die wenigen Freeloader, die trotzdem das tokenbasierte Netz nutzen, werden sich auf diese uneigennützigen Provider Peers beschränken.

Die meisten Internetnutzer haben bei ihren Internetanbindungen eine deutlich geringere Uploadrate als Downloadrate. Im Durchschnitt ist der Upload etwa viermal langsamer. Aus diesem Grunde sind die Peers kaum in der Lage, so viel ins Netz hinaufzuladen, wie sie Content beziehen wollen. Je nach dem, ob sie bereit sind, den Client wesentlich länger laufen zu lassen, als die Downloads benötigen würden, werden sie stets knapp bei Kasse sein, und sich durch die Architektur mit den Tokens eingeschränkt fühlen. Ob dieser Nachteil genügend durch die schnelleren Downloads kompensiert wird, hängt von jedem einzelnen Nutzer ab.

Wenn das tokenbasierte Peer-to-Peer Netzwerk im Internet zum Einsatz kommt, wird das Pricing taktisch nicht mehr so wichtig sein, wie im Design erwartet. Die meisten Benutzer werden den Client einmal laufen lassen, einige Dateien zum Upload anbieten und andere Dateien suchen. Wenn neuer Content im Peer-to-Peer Netzwerk angeboten wird, ist es zwar möglich, dass einzelne Anbieter anfangs dafür mehr verlangen. Die grosse Mehrheit der Benutzer wird aber warten, bis sich dieser Content verbreitet und preislich in dieselbe Region kommt, in welcher sich die anderen Dateien auch bewegen. Der Client unterscheidet sowieso nicht zwischen verschiedenen Dateien und versucht alles ins Internet zu verkaufen, was ihm im Sharedfolder unter den Lesekopf kommt. So wird es eine Kleine Zahl von Nutzern geben, die versuchen möglichst in Kurzer Zeit an bestimmte Downloads zu kommen oder taktisch möglichst viele Tokens zu verdienen. Der Rest der Benutzer wird wahrscheinlich kaum viel an der Grundkonfiguration herumschrauben, sondern einfach einmal ein paar Files anbieten, und andere herunterladen, sobald sich eine gewisse Anzahl Tokens angesammelt hat. Die Absicht der meisten Benutzer wird sein, auf eine so einfache Art, wie möglich Dateien herunterzuladen, ohne sich um die technischen Details kümmern zu müssen. So wird die Auction-basierte Methode eher ein Schattendasein fristen, falls nicht die Clients von den Benutzern dazu veranlasst werden, aktiv spekulative Downloadangebote an die anderen Peers zu senden, bevor die Datei bei Nichterfolg nach einiger Zeit zu den vorgegebenen Preisen heruntergeladen wird.

Die Tokens werden sowieso nur denjenigen Peers zum Problem, die mit möglichst wenigen Uploads versuchen, ihre Downloads zu finanzieren. Peers, die zwar Dateien anbieten, aber nicht exzessiv Content herunterladen, werden nie einem Mangel an Tokens zu befürchten haben und deshalb nicht darauf achten müssen, ob jetzt eine Datei irgendwo im Peer-to-Peer Netzwerk 10% günstiger zu haben ist. Man kann sich das Ganze etwa so vorstellen, dass des Filesharing Netzwerk etwa ähnliche Eigenschaften hat wie Netzwerk ohne interne Währung, einfach dass die verschiedenen Applikationen im Hintergrund trotzdem den Transfer abrechnen.

Ob die Tauschbörsen an sich eine Zukunft haben, muss sich auch noch zeigen. Zu gross ist die Versuchung, dass die Benutzer urheberrechtlich geschütztes Material tauschen. Wahrscheinlich werden im tokenbasierten Filesharing Netzwerk einige Leute denken, dass es legal sei, wenn sie

dem ganzen Geschehen teilnehmen, das sie selber für den heruntergeladenen Content bezahlt haben. Dies wird natürlich Firmen af den Plan rufen, die rechtliche Schritte gegen das Geschehen unternehmen werden. Da es aber sehr schwierig ist, den gesamten Content irgendwie zu filtern oder trennen, wird stattdessen einfach das ganze Peer-to-Peer Netzwerk ausser Betrieb gesetzt.

Zurzeit ist das tokenbasierte Netzwerk noch nicht dafür ausgelegt, um Multimediafiles offiziell kommerziell zu verteilen. Für diesen Zweck müsste das Protokoll auf diese Weise so geändert werden, dass die Urheber der Dateien am Tausch verdienen, und nicht diejenigen, die ein File gerade Uploaden.

9. Ausblick

In der heutigen Zeit wird es mehr und mehr populär, ganze Filme via Peer-to-Peer Netzwerke zu tauschen. Eine einzige solche Datei beansprucht je nach Internetanbindung die volle Bandbreite über mehrere Stunden, sofern Peers vorhanden sind, welche die Datei mit der dementsprechend hohen Geschwindigkeit liefern können.

In diesem Falle ist es wichtig, die Last innerhalb eine Peer-to-Peer Netzwerks auszubalancieren, da sonst die wenigen Provider Peers hoffnungslos überlastet sind und nur sehr wenige Customer Peers mit ihrem Content befriedigen können.

Bedarf einer Lastbalancierung in der Zukunft

Auch wenn in Zukunft die durchschnittliche Bandbreite der Peers stetig zunimmt, ist es kaum anzunehmen, dass die Grösse der getauschten Dateien im gleichen Masse anwächst. Eine Videodatei ist von den Multimediadateien diejenige mit am meisten Datendurchsatz Pro Zeiteinheit, und da ist kaum anzunehmen, dass sich der Mensch in Zukunft mit viel rechenintensiveren Multimediafiles berieseln lässt, wenn man mal von der Qualitätsverbesserung in Bild und Ton absieht. Auf der anderen Seite verbessern sich die Kompressionsverfahren stetig, so dass Videoströme bald einmal wesentlich schneller, als in Echtzeit übers Internet geschickt werden können.

Wenn die durchschnittliche Bandbreite stetig zunimmt, aber die Dateigrösse nicht mehr anwächst, wird in Zukunft die Internetanbindung im Durchschnitt wieder weniger ausgelastet, so dass es vielleicht eine Zeit geben wird, in welcher Dateien im Gigabytebereich in Minutenbruchteilen übertragen werden können, was eine aktive Lastbalancierung überflüssig macht.

Erweiterungsmöglichkeiten für das Projekt

In dem vorliegenden Projekt handelt es sich um die erste Version eines neuen Designs. Je nach dem, wie sich die Tokenbasierte Lastbalancierung in der Realität bewährt, sollten vielleicht einige Parameter angepasst werden. Je nach dem, wie kommerziell die Benutzer des Peer-to-Peer Netzwerks denken, wird die Währung nur als notwendige Gegenleistung für Dateien betrachtet oder aber als Mittel um damit zu Spekulieren benutzt.

Es wird sich zeigen, wie gross der Anteil an altruistischen Peers oder beinahe Freeloader im Netzwerk sein wird und wie sich diese Werte weiter entwickeln werden. Mit den erhobenen demographischen Daten können dann die Algorithmen im System verfeinert werden. Es besteht generell der Trend, dass der Client dem Benutzer immer mehr Arbeit abnimmt, so dass der Benutzer sich kaum mehr um die Preisverhandlungen kümmern muss.

In dieser Arbeit wurde noch nicht behandelt, wie genau der Preis einer Datei ermittelt wird, falls der Benutzer nur Teile davon herunterlädt, oder einen Download auf mehrere Provider Peers verteilt wird. Es wurde generell angenommen, dass eine Datei im Vergleich teurer ist, wenn der Provider Peer bereit ist, sie mit hohem Durchsatz zu liefern. Wie steht es aber, wenn der Customer Peer Fragmente gleichzeitig von vielen Peers mit niedrigerer Bandbreite anfordert, so dass sich diese zu einem grossen Durchsatz aufaddieren? Wenn alle Peers so handeln, verdienen die Provider Peers im Vergleich weniger, obwohl der Durchsatz gleich gross ist und der für die Preisabsprachen nötige Overhead sogar anwächst.

Wenn der Bedarf besteht, könnten auch noch weitere Funktionen implementiert werden, die man aus der Wirtschaft und Börse kennt. Ein Beispiel wäre eine Downloadoption, mit welcher die Peers gegen ein kleines Entgelt sich das Recht erkaufen, in einem bestimmten Zeitrahmen eine Datei zu einem vorher festgelegten Preis zu kaufen.

Eine andere Idee wären virtuelle Versicherungen, bei welchen sich die Peers gegen abgebrochene Downloads, schlechten Content oder Betrug versichern können. So brauchen sie nur ein kleines Entgelt an einen dafür zuständigen Peer zu zahlen, welcher hauptsächlich damit beschäftigt ist, die verschiedenen Peers nach ihrer Reputation zu bewerten, und daraus den Preis für die passenden Versicherungspolicen zu ermitteln.

Momentan läuft der Trend in Richtung paralleler Nutzung von verschiedenen File-sharing Architekturen. Für diesen Zweck werden immer mehr Applikationen implementiert, welche sich mit mehreren Peer-to-Peer Netzwerken gleichzeitig verbinden können. Der Community bringt das insofern einen Nutzen, als der Content dementsprechend besser unter den Netzwerken durchmischt wird. So ist es durchaus möglich, dass das tokenbasierte Filesharing Netzwerk sich gleichzeitig mit anderen Peer-to-Peer Systemen verbindet, und von dort Content bezieht oder zur Verfügung stellt. Simultanes Downloaden von mehreren verschiedenen Systemen gleichzeitig wird in naher Zukunft noch nicht möglich sein, da die für die Identifikation benutzten Hashwerte zueinander nicht kompatibel sind.

Wie schon unter [3.2.3] erwähnt, könnten die Tokens zudem als virtuelles Zahlungsmittel für Dienste wie Rechenkapazität, Speicherplatz oder andere Ressourcen dienen.

Wenn die Antwortzeiten zwischen den Peers ein gewisses Mass nicht überschreitet, ist es zudem Möglich, das Tokenbasierte Filesharing Netzwerk für Livestreams zu verwenden. Die Streams werden von Peer zu Peer Baumförmig weitergegeben. Wenn jemand den Stream konsumiert aber gleichzeitig an einen weiteren Peer weiterleitet, so wird die Bilanz ziemlich ausgeglichen sein. Peers, welche den Stream an mehrere Customer weitersenden, werden für ihren Service verdienen, wo hingegen reine Konsumenten dafür bezahlen müssen. Ob dies gewisse Vorteile gegenüber dem Internet Broadcasting bringt, bei welchem die Router die Datenströme für die Endbenutzer gratis verteilen, wird sich dann zeigen müssen.

10. Referenzen

[1] S. Kamvar, B. Yang, and H. Garcia-Molin. *Addressing the Non-Cooperation Problem in Competitive P2P Systems.* Workshop on Economics of Peer-to-Peer Systems 2003, Berkley, CA.
http://www.sims.berkeley.edu/research/conferences/p2pecon/papers/s7-kamvar.ps

[2] Webseite der Java Programmierumgebung *Eclipse.*
http://www.eclipse.org

[3] E. Adar and B. Hberman. *Free Riding on Gnutella.*
http://www.firstmonday.dk/issues/issue5_10/adar/index.html, 2000.

[4] Gnutella Wedsite: http://www.gnutella.com;

[5] Gnutella2: Überarbeitete Version des Gnutella Protokolls: http://www.gnutella2.com
Information: http://www.infoanarchy.org/wiki/wiki.pl?Gnutella2

[6] Hilfe zu eMule. Open Source Projekt basierend auf eDonkey Protokoll. *eMule.chm*
http://www.emule-project.net/

[7] K. Lai, M. Feldman, I. Stocia, and J. Chuang. *Incentives for Cooperation in Peer-to-Peer Networks.* Workshop on Economics of Peer-to-Peer Systems 2003, Berkley, CA.
http://www.sims.berkeley.edu/research/conferences/p2pecon/papers/s1-lai.pdf

[8] P. Golle, K. Leyton-Brown, and I. Mironov. *Incentives for Sharing in Peer-to-Peer Networks.* Computer Science Department Stanford University, 1999.

[9] V. Vishnumurthy, S. Chandrakumar, and E. Gün Sirer. *KARMA: A Source Economic Framework for Peer-to-Peer Recource Sharing.* Workshop on Economics of Peer-to-Peer Systems 2003, Berkley, CA.
http://www.sims.berkeley.edu/research/conferences/p2pecon/papers/s5-vishnumurthy.pdf

[10]Mojo Nation: http://www.openp2p.com/pub/a/p2p/2001/01/11/mojo.html

[11]Dreamtech. *Peer-to-Peer Applikationen entwickeln.*
mitp-Verlag, Bonn, 2002. ISBN 3-8266-0832-1

[12]D. Hausheer, N. Liebau, A. Mauthe, R. Steinmetz, B. Stiller: *Token-based Accounting and Distributed Pricing to Introduce Market Mechanisms in a Peer-to-Peer File Sharing Scenario*; ETH Zurich / TU Darmstadt, April 1, 2003.

[13]T. Moreton and A. Twigg. *Trading in Trust, Tokens, and Stamps.* Workshop on Economics of Peer-to-Peer Systems 2003, Berkley, CA.
http://www.sims.berkeley.edu/research/conferences/p2pecon/papers/s2-moreton.pdf

Institut für
Technische Informatik und
Kommunikationsnetze

Prof. Dr. Burkhard Stiller, ETZ G61.4, 8092 Zürich, Tel. +41 1 632 7016, Fax +41 1 632 1035, E-Mail stiller@tik.ee.ethz.ch

Diplomarbeit

für

Herrn Reto Keiser

Aufgabenstellung:	Dipl. El.-Ing. ETH David Hausheer
Thema:	**Distributed Content-based Accounting and Charging in P2P**
Beginn der Arbeit:	13.05.2003
Abgabetermin:	12.09.2003
Betreuung:	Dipl. El.-Ing. ETH David Hausheer, Dipl. Inf.-Ing. ETH Pascal Kurtansky
Arbeitsplatz:	Im ETZ, C96
Hilfsmittel:	Computer mit Standardinstallation

1. Einleitung

Peer-to-peer (P2P) Netzwerke abstrahieren von der darunterliegenden Netzwerkinfrastruktur wie z.B. dem Internet und stellen auf Applikationsebene skalierbare und effiziente Routingmechanismen zur Verfügung. Mittels Replikation von Ressourcen wie z.B. Content können P2P-basierte Systeme eine viel höhere Robustheit und Performance erreichen als traditionelle Client/Server-basierte Applikationen und zeichnen sich so durch eine hohe Verlässlichkeit aus. Filesharing Systeme wie Gnutella, KaZaA, und E-Donkey verwalten auf diese Weise grosse Mengen an Content. Da Benutzer von P2P Systemen allerdings keinen direkten Nutzen daraus ziehen können, wenn sie anderen Benutzern ihre eigenen Ressourcen zur Verfügung stellen, verhalten sich nur wenige von ihnen entsprechend kooperativ. Viele (sogenannte Free-loader) konsumieren nur Content von anderen, aber tragen selber nichts zum System bei. Das hat zur Folge, dass die Gesamtperformance des Systems sinkt und die guten Eigenschaften eines P2P Systems wie z.B. Load Balancing nicht erreicht werden. Des weiteren ist die Qualität des bereitgestellten Content häufig schlecht, bzw. existiert meistens im Voraus keine verlässliche Information darüber. Mit den bisher entwickelten Filesharing Systemen ist es zudem nicht möglich, bezahlten Content auszutauschen, da die dazu nötigen Abrechnungs- und Sicherheitsmechanismen fehlen bzw. nicht implementiert sind. Diese wären aber für eine kommerzielle Verwendung der P2P Technologie Voraussetzung. Im Vergleich zu zentralisierten Lösungen sind verteilte Charging- und Accountingmechanismen allerdings viel schwieriger zu implementieren, und es kann leicht zu Missbräuchen kommen, da es keine zentral gesteuerte Kontrolle über das System gibt.

2. Aufgabenstellung

Das Ziel dieser Diplomarbeit ist es, basierend auf den in der Einleitung diskutierten Überlegungen und entsprechenden bereits vorhandenen Ideen und Lösungen ein Konzept für die Abrechnung von Content in P2P Systemen zu entwickeln und dieses anhand eines zu implementierenden Prototypen zu evaluieren. Das Konzept soll generell für verschiedene Szenarien verwendbar sein, im speziellen aber die besonderen Aspekte von Content wie z.B. dessen Kopierbarkeit berücksichtigen.

Literaturrecherche und Begriffsdefinition

Um einen Überblick über das Forschungsgebiet und die konkrete Problemstellung zu erhalten, sollen vorhandene Literatur, Konzeptideen und Tools in den Bereichen Content Pricing und Charging, Distributed Accounting und Peer-to-peer untersucht werden. Dabei sind sowohl kommerzielle als auch nicht-kommerzielle Lösungen von Interesse. Die verschiedenen Konzepte sind zu klassifizieren und die dabei verwendete Terminologie genau zu definieren und abzugrenzen.

Konzept und Design

Basierend auf den untersuchten Ideen und der Literatur soll ein Konzept entwickelt werden, welches das fokussierte Szenario konkretisiert und die behandelten Aspekte genau beschreibt. Diese sollen gegenüber weiterführenden Überlegungen, welche den Rahmen der Arbeit sprengen würden, abgegrenzt werden. Designalternativen müssen zwar aufgezeigt und diskutiert werden, für die Implementation soll dann aber ein einziger Designansatz ausgewählt werden.

Konkret soll das zu entwickelnde Design im Sinne von [3] einen verteilten Price Setting und Dissemination Mechanismus für Content Ressourcen beinhalten, welcher es dem Benutzer eines P2P Filesharing Systems ermöglichen soll, Preise für Content festzulegen bzw. aufgrund eines Preises zu entscheiden, bei welchem Provider bestimmter Content bezogen und wie abgerechnet werden soll. Unter anderem sollen dabei Sicherheitsaspekte (Price Authenticity and Integrity) und Verteiltheitsaspekte (Price Caching, Content Splitting) berücksichtigt werden. Die gewählte Beschreibung von Preisen soll sich zudem leicht in den Content selbst einbinden lassen und effizient zur verteilten Abrechnung für die Benutzung von Content verwendbar sein.

Implementation und Evaluation

Das entwickelte Design soll schliesslich in einem Prototypen in Java implementiert werden. Die Implementation innerhalb eines bestehenden Framework (z.B. JXTA) ist denkbar, muss aber auf jeden Fall zuerst mit dem Betreuer abgesprochen werden. Der Prototyp soll eine Evaluierung des Systems aufgrund von messbaren Grössen ermöglichen. Die Evaluation soll Stärken und Schwächen des Designs und der Implementation aufzeigen und dient zugleich als Proof-of-Concept. Anhand von Simulationen und analytischen Überlegungen soll überprüft werden, ob die Resultate aus den real-world Experimenten mit dem Prototypen richtig sind.

3. Vorgehensweise

Folgende Schritte beschreiben einen sinnvollen Vorschlag für den Ablauf der Arbeit:

- Machen Sie sich ausgehend von den angegebenen Literaturhinweisen mit den Themen der Arbeit vertraut.
- Suchen Sie nach weiteren Arbeiten in dem behandelten Forschungsbereich. Verwenden

Sie dazu Bibliotheken und vor allem auch das Internet. Dabei sind besonders Veröffentlichungen im Rahmen von Konferenzen und Workshops von Interesse. Diese sollten nur in Ausnahmefällen aus dem Zeitraum älter als zwei Jahre sein. Machen Sie sich auf diese Art und Weise mit existierenden Ansätzen vertraut und dokumentieren Sie diese von Anfang an in ihrem Bericht.

- Ausgehend von den damit ermittelten Grundlagen erstellen Sie im Sinne der Aufgabenstellung einen eigenen Anforderungskatalog und treffen die Annahmen mit welchen Sie Ihre Aufgabe klar gegenüber möglichen Designalternativen abgrenzen. Beschreiben Sie auch das gewählte Szenario und das zugrundeliegende System auf welchem Sie Ihre Arbeit basieren.
- Zeigen Sie, wie sich Ihre Anforderungen in Form von Methoden und Modulen umsetzen lassen. Implementieren Sie einen Prototypen, anhand dessen Sie Ihre Konzepte veranschaulichen und deren Funktionsweise zeigen können. Bei der Implementierung ist auf eine möglichst modulare Aufbauweise des Programmiercode zu achten, um eine spätere Weiterverwendung zu erleichtern. Wo verfügbar sollten auch bereits vorhandene Codefragmente und Standards wenn möglich wiederverwendet werden.
- Anhand von Simulationen und Experimenten mit dem von Ihnen entwickelten Prototypen evaluieren Sie anschliessend Ihre Konzepte anhand messbarer Vergleichgrössen.
- Abschliessend muss die Arbeit beschrieben und zusammengefasst werden. Dabei ist es besonders wichtig, noch offene Punkte aufzuzeigen und mögliche Lösungen zu skizzieren.

4. Bemerkungen

- Nach Ablauf der ersten Woche muss ein Zeitplan für den Ablauf der Arbeit eingereicht und mit den Betreuern diskutiert werden.
- Etwa zwei Monate nach Beginn der Arbeit findet eine kurze Zwischenpräsentation mit dem Assistenten und dem betreuenden Professor statt.
- Am Ende der Arbeit muss ein schriftlicher Bericht eingereicht werden, der die geleistete Arbeit dokumentiert. Dieser Bericht sollte so geschrieben sein, dass er für einen Nicht-Spezialisten verständlich ist. Des weiteren muss er alle wichtigen Vorbedingungen, Definitionen und Design-Entscheidungen enthalten.
- Es muss ein regelmässiger Kontakt (möglichst mindestens einmal pro Woche) zwischen dem Studenten und seinen Betreuern erfolgen, per Telefon, E-Mail, Treffen oder auf anderem Wege. Diese Kontakte sollen dazu verwendet werden, den Fortgang der Arbeit darzustellen und auftretende Probleme zu diskutieren.
- Besonders wichtig ist das regelmässige (möglichst tägliche) Lesen von E-Mails.

5. Ergebnisse der Arbeit

Die Arbeit muss innerhalb eines 20minütigen Vortrags am TIK vorgestellt werden. Der genaue Termin dieses Vortrags wird noch bestimmt werden, voraussichtlich wird er jedoch Ende August / Anfang September 2003 sein. Abgesehen vom Vortrag müssen folgende Dokumente eingereicht werden:

- Der in deutsch oder englisch abgefasste Bericht. Dieser Bericht muss folgende Punkte beinhalten: Eine Beschreibung des untersuchten Forschungsgebiets und der untersuchten Konzepte und Tools, eine Begründung für die Wahl der ausgewählten Anforderungen sowie der Design-Varianten, eine Liste von gelösten und ungelösten Problemen, Inhalts-,

Tabellen- und Bildverzeichnisse, Literaturangaben und eventuell Anhänge (z.B. Programmcode, Glossar und ähnliches). Der Bericht muss mit einer Beurteilung abschliessen, in wie weit die ursprüngliche Aufgabenstellung erreicht und der aufgestellte Zeitplan eingehalten werden konnten. Es müssen insgesamt fünf gebundene und doppelseitig gedruckte Kopien des finalen Berichts eingereicht werden. Der Bericht sollte wenn möglich in Framemaker erstellt werden, um eine eventuelle spätere Weiterverwendung zu erleichtern.

- Die Zusammenfassung sollte wie folgt gegliedert sein:

  Einleitung - Ziele - Ergebnisse - Offene Punkte
- Ein englisches oder deutsches (jeweils die Sprache, in der der Bericht NICHT geschrieben ist) Abstract von ein bis zwei Seiten Länge. Dieses sollte einen schnellen Überblick über die gesamte Arbeit ermöglichen. Es muss hinter der ersten weissen Seite in den Bericht eingefügt werden.
- Eine elektronische Version des Berichts und alle sonstigen erstellten Dokumente (z.B. Modelle). Bilder sollten dabei zusätzlich als gesonderte Dateien vorliegen, in einem gut verarbeitbaren Format (z.B. EPS). Die Abgabe sollte entweder auf einer CD oder durch ein seperates Verzeichnis im für die Arbeit angelegten Account erfolgen.
- Sämtliche referenzierte und bearbeitete Literatur, sowohl in elektronischer als auch in ausgedruckter Version.

6. Literaturangaben

[1] R. Buyya, S. Vazhkudai: *Compute Power Market: Towards a Market-Oriented Grid;* IEEE Session on Global Computing on Personal Devices (In conjunction with CCGRID 2001), Brisbane, Australia, May 2001.

[2] R. Dingledine, M. J. Freedman, D. Molnar: *Accountability*; In Peer-To-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly & Associates, Chapter 16, pp. 217 - 340, 1st edition, March 15, 2001.

[3] D. Hausheer, N. Liebau, A. Mauthe, R. Steinmetz, B. Stiller: *Token-based Accounting and Distributed Pricing to Introduce Market Mechanisms in a Peer-to-Peer File Sharing Scenario*; ETH Zurich / TU Darmstadt, April 1, 2003.

[4] W. Thigpen, T. J. Hacker, L. F. McGinnis, B. D. Athey: *Distributed Accounting on the Grid*; In Proceedings of the 6th Joint Conference on Information Sciences, pp.1147-1150, 2002.

[5] H. Varian: *Markets for Information Goods*, In Proceedings of Monetary Policy in a World of Knowledge-Based Growth, Quality Change, and Uncertain Measurement, June 1998.

Die Suche nach weiterer Literatur wird sehr empfohlen.