Freenet

Ein anonymes, dezentrales Filestorage System

Alexander Hausner
Alexander.Hausner@informatik.stud.uni-erlangen.de

01.07.2002

Kurzer Überblick

Dieser Artikel beschreibt das Peer-to-Peer System Freenet. Ein besonderes Merkmal an Freenet ist die komplette Anonymität. Das heisst, dass man ähnlich zu Freehaven [5] keine relevanten Daten, sowohl über den Informationsanbieter als auch über den Informationskonsumenten, herausfinden kann. Ausserdem erfolgt die Datenübertragung verschlüsselt. Das Konzept von Freenet basiert auf Ian Clarke's Diplomarbeit [3], der noch immer für die Weiterentwicklung von Freenet mitverantwortlich ist. Sein Ziel ist es, ein nicht-zensierbares Datenspeicher-System bereitzustellen.

1 Motivation

1.1 Informationsfreiheit

Viele Menschen sehen das Internet als ein Informationssystem, in dem man einerseits seine eigene Meinung frei vertreten kann, andererseits Informationen abrufen kann und dabei anonym bleibt. Das ist ganz klar nicht der Fall, da der meiste Datenverkehr unverschlüsselt erfolgt und es leicht möglich ist, ihn abzuhören. Es ist ein offenes Geheimnis, dass US-Geheimdienste schon längst den E-Mailverkehr überwachen.
Von staatlicher Seite aus werden in Ländern wie China oder manchen arabischen Staaten die übertragenen Daten nach Inhalten gefiltert, um die Bevölkerung vor der drohenden Amerikanisierung zu „schützen“ oder einfach, weil es den vorherrschenden religiösen Ansichten nicht entgegenkommt.
Doch soweit muss man gar nicht gehen. Selbst in Deutschland wurden immer wieder Website-Betreiber für ihre Inhalte verantwortlich gemacht: Man hat die Provider dazu gezwungen, die Daten - mp3's, warez, rechtsextreme Inhalte - zu löschen.

1.2 Ein dezentrales System

Für ein dezentrales System sprechen mehrere Aspekte. Als erstes wäre natürlich Ausfallsicherheit und Fehlertoleranz zu nennen. Wenn ein einfacher Computer - sozusagen ein zentrales System - einen Hardwaredefekt hat, wird dieser nicht mehr oder nur sehr eingeschränkt funktionsfähig sein, je nachdem ob zum Beispiel nur die Netzwerkkarte kaputt ist oder gleich der Prozessor. Dasselbe Prinzip kann man auf das Internet ausweiten: Fällt ein Webserver aus, so sind dessen Seiten vorerst nicht mehr abrufbar; natürlich unter der Annahme, dass man sie nicht auf andere Server gespiegelt hat.

Ein weit häufigeres Problem als Serverausfälle sind Verbindungsprobleme. Nehmen wir ein extremes Beispiel an, welches so noch nicht vorgekommen ist: Der größte Teil des deutschen Internetverkehrs wird über das Frankfurter Backbone abgewickelt. Sollte es dort zu einem Ausfall kommen, würde fast der gesamte deutsche Internetverkehr zusammenbrechen. Um die Wahrscheinlichkeit für so ein Vorkommnis zu vermeiden, setzt man im allgemeinen redundantere Hardware ein.
Ian Clarke geht jedoch von einer korrupten Welt aus und sieht die große Gefahr in dem Machtpotenzial, das Unternehmen im Besitz solcher zentralen Systeme ausüben könnten, um Vorteile für sich herauszuziehen. Beim Frankfurter Backbone mag das weniger kritisch erscheinen, doch wie sieht es denn mit der Vergabe von Domain-Namen bzw. IP-Adressen aus? Seit Oktober 1998 ist die dafür zuständige Dachorganisation die ICANN[1]. Es ist durchaus denkbar, dass aufgrund politischer Konflikte oder durch Korruption bestimmte Staaten bzw. Unternehmen bei der Vergabe der IP-Adressen benachteiligt werden und somit keine Gleichberechtigung gewährleistet ist. Ähnliches ist bei der Domainvergabe schon passiert. Einige Personen haben auf gut Glück bestimmte Domainnamen bekannter Firmen für sich reservieren lassen, um sie dann möglichst gewinnträchtig verkaufen zu können.

2 Freenet

2.1 Allgemeine Architektur

Um die im vorangegangenen Abschnitt besprochenen Probleme vermeiden zu können, war es naheliegend, Freenet als völlig dezentrales Peer-to-Peer System zu entwerfen. Völlig dezentral heisst in diesem Fall, dass es, im Gegensatz zu eDonkey [6] oder Fasttrack [7], ohne einen sogenannten *Vermittlungsserver* auskommt. Weiterhin muss absolute Anonymität der Benutzer gewährleistet sein, egal ob als Anbieter von Information oder als Konsument. Es werden also nicht nur die Daten selbst verschlüsselt, um sich vor Inhaltsfiltern zu schützen, sondern auch alle Informationen über einen Knoten. Das heisst im Einzelnen, dass man keine Aussage über die Herkunft eines Benutzer treffen kann, mit wem er kommuniziert und welche Daten er anbietet.
Man ging also von einem völlig paranoiden Ansatz aus: Einem Peer-to-Peer Dienst kann man grundsätzlich nicht vertrauen, jeder Knoten könnte bösartig sein und beliebige Knotenausfälle müssen ohne Datenverlust toleriert werden.
Freenet ist ausserdem mehr als nur ein reiner Dateitauschdienst. Da die Architektur der Knoten ziemlich allgemein gehalten ist, ist es mit Zusatztools möglich, in Newsgroups zu diskutieren, E-Mails zu schreiben und sogar Internetseiten anbieten, die mit einem normalen Browser angesehen werden können.

2.1.1 GUID Keys

Jede Datei in Freenet wird eindeutig über Schlüssel identifiziert, die sogenannten *Global Unique Identifier*-Keys (GUID-Key). Erzeugt werden diese Schlüssel mit dem Secure-Hash-Verfahren (SHA-1), das als absolut sicher und kollisionsfrei eingestuft wird. Es gibt verschiedene Arten von Schlüsseln:
Der einfachste Schlüsseltyp ist der *Content-Hash-Key* (CHK), der - wie der Name schon sagt - aus dem Inhalt der Datei erzeugt wird. Sollten zwei User exakt die gleiche Datei einbringen,

[1] „Internet Corporation for Assigned Names and Numbers“. Die ICANN ist ein unabhängiges Unternehmen und hegt keinerlei profitables Interesse. Die Gründung dieser Organisation löste damals die IANA (Internet Assignment Numbers Authority), die zur US-Behörde gehörte, ab.

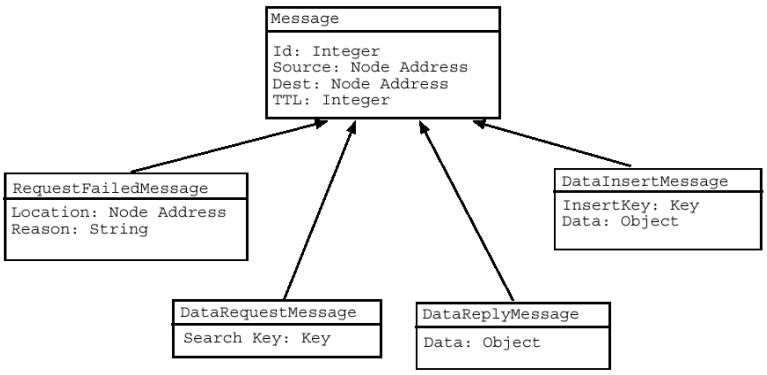
so werden auch die resultieren Hashwerte genau gleich sein. Vergleichbar mit einer URL[2] im World Wide Web, zeigt in Freenet ein CHK auf die richtige Datei.
Ein komplexerer Schlüsseltyp ist der *Signed-Subspace-Key* (SSK). Mit ihm ist es möglich, den eigenen Dateien einen Namensbereich zuzuordnen, den jeder lesen kann, aber nur der Besitzer auch schreiben.
Ein SSK für einen Namensbereich besteht aus einem zufällig gewähltem public / private Schlüsselpaar. Wenn der Benutzer nun eine Datei einbringen will, wählt er als erstes eine kurze Beschreibung der Datei. Der public-Teil des Schlüsselpaares und die Beschreibung werden unabhängig voneinander gehashed, danach zusammen mit XOR verknüpft und wieder gehashed. Mit dem private-Teil des Schlüsselpaares wird eine Signatur der Datei erzeugt, um einen Integritätscheck bereitzustellen. Die anderen Knoten können nun mit dem public-Teil des Schlüssels die Datei verifizieren, bevor sie sie akzeptieren.
Um eine Datei zu bekommen, braucht ein Benutzer nur die Beschreibung der Datei und den public-Teil des Schlüssels. Um andererseits eine Datei einzubringen und die richtige Signatur zu erzeugen, braucht man den private-Teil des Schlüssels.
In den meisten Fällen wird ein SSK benutzt, um den CHK einer Datei zu speichern, um mit diesem schliesslich die Datei zu bekommen. Dieses Vorgehen ist vergleichbar zu der Suche in einem Stichwortverzeichnis, in dem man über die angegebene Seitenzahl den eigentlichen Artikel findet.
Es sind aber auch andere Szenarien mit dem SSK denkbar. So könnte man beispielsweise auch ein regelrechtes (Pseudo)-Domain-Name-System aufbauen, in dem man den Public-Key eines Knoten speichert. Damit ist es möglich, selbst bei wechselnden IP-Adressen, immer den richtigen Knoten zu finden.

2.1.2 Nachrichten

Es gibt vier verschiedene Nachrichtentypen: *Data Request*, *Data Reply*, *Request Failed* und *Data Insert*. Alle Nachrichten tragen eine eindeutige ID, einen TTL[3] und je nach Nachrichtentyp entweder einen Key (bei *Data Request*), ein Object (bei *Data Reply*) oder beides (bei *Data Insert*). *Request Failed* enthält einen ErrorString. Die Nachrichtenhierarchie ist am Ende des Abschnitts grafisch veranschaulicht.
Beim Passieren eines Knotens werden die Source- und Destination-Adressen jedesmal mit denen des aktuellen bzw. nächsten Knotens ausgetauscht, was Freenet grundsätzlich von anderen Peer-to-Peer Systemen unterscheidet: Es findet beim Datenaustausch niemals ein direkter Verbindungsaufbau zwischen zwei entfernten Knoten statt. Statt dessen wandern die Nachrichten von einem benachbarten Knoten zum Nächsten, damit niemals nachvollzogen werden kann, welche User gerade miteinander kommunizieren. Das bedeutet sogar, dass ein Knoten niemals weiss, ob der Knoten, von dem er eine Nachricht bekommen hat, der Initiator war, und umgekehrt, ob der Knoten, an dem eine Nachricht verschickt wurde, der Empfänger ist.

[2]URL steht für „Uniform Resource Locator“. Eine URL kennzeichnet die Adresse eines jeden Dokumentes im Internet eindeutig.

[3]TTL ist die gebräuchliche Abkürzung für „Time to Live“. Jedesmal, wenn eine Nachricht einen Knoten passiert, wird der Wert um eins erniedrigt. Sinkt der TTL auf 0, so verwirft der Knoten die Nachricht. Damit verhindert man, dass Nachrichten, die ihr Ziel nicht finden, „ewig“ im Netz bleiben.

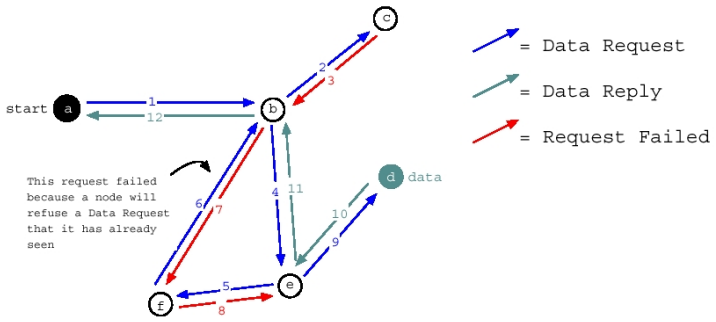


2.1.3 Routing

Wie eingangs schon erwähnt, muss Freenet ohne einen Vermittlungsserver auskommen, denn dieser hätte den Nachteil, dass er als zentrales System gezielt angreifbar wäre und ein Ausfall das Netz lahm legen würde.
Gnutella [4] hat auch keinen Indexserver und sendet deshalb Broadcasts an eine bestimmte Anzahl erreichbarer Knoten, was zu einem sehr hohen Nachrichtenaufkommen führt und das Netz verlangsamt. Bei Freenet dagegen versucht man einen Mittelweg zu gehen. Jeder Knoten schickt eine Nachricht an den Knoten weiter, von dem er glaubt, dass dieser einen Schlüssel besitzt, der dem Gesuchten am nächsten ist. Diese Vorgehensmethode wird als *steepest-ascent hill-climbing search* [3] bezeichnet.

2.1.4 Daten anfordern

Wie werden nun Daten gefunden? Jeder Knoten hält eine Routingtabelle, in der die GUID-Keys der Nachbarknoten eingetragen sind. Sollte nun eine Anfrage eintreffen, so prüft der Knoten erst seine eigenen Daten. Ist der Schlüssel enthalten, so schickt er ein *Data Reply*. Falls der Schlüssel nicht enthalten ist, so schickt der Knoten an *Data Request* Nachricht an denjenigen Knoten weiter, der einen Schlüssel besitzt, welcher dem Gesuchten am nächsten ist. Diese Kette setzt sich solange fort, bis entweder der TTL auf 0 gesunken ist oder die Datei gefunden wurde. Nun geht die ganze Prozedur rückwärts. Alle Knoten in der Kette schicken *Data Failed* oder *Data Reply* bis zum Initiator zurück.
Im ersten Fall hat der User nun die Möglichkeit, die Suche mit einem höheren TTL erneut zu versuchen.
Im zweiten Fall enthält die *Data Reply* Nachricht die gesuchte Datei. Auf dem Weg zum Initiator speichert jeder Knoten eine Kopie der Datei und identifiziert sich selbst als Besitzer, so dass der nachfolgende Knoten ihn in seine Routingtabelle einträgt. Auch hier zeigt sich wieder das Bestreben nach Anonymität: Man kann nicht herausfinden, welcher Knoten die Datei ursprünglich besaß.
Diese Vorgehensweise wird als *aggressive caching* bezeichnet und es gab Diskussionen, ob es sinnvoll ist, soviel Ressourcen zu verschwenden. In der aktuellen Implementierung von Freenet wird nach dem Zufallsprinzip ausgewählt, ob ein Knoten eine Datei zwischenspeichert. Die Anonymität bleibt hier genauso gewährleistet, es wird weniger Speicher verschwendet und mathematisch gesehen ist das Verfahren genauso gerecht wie das aggressive caching.



Die obige Grafik zeigt ein Beispiel wie so ein Nachrichtenablauf aussehen könnte. Knoten A stellt eine Anfrage an B, welcher wiederrum *Data Request* an Knoten C weiterleitet (1-2). Dieser Knoten ist der letzte in der Kette und antwortet mit *Data Failed*. Knoten B probiert nun die zweitbeste Alternative laut seiner Routingtabelle und schickt das *Data Request* an E, E an F und Knoten F schliesslich wieder zu B weiter (4-6). Hier kommt es nun zu einem Zyklus. Knoten B weiss, dass er diese Request Nachricht bereits verschickt hat und antwortet deswegen an F mit *Request Failed* (7). Diese Nachricht wird bis zum Auslöser des Zykluses zurückgeschickt, im gegebenen Fall also Knoten E, der nun die nächstbeste Möglichkeit probiert. Im Beispiel besitzt Knoten D die gesuchte Datei und *Data Reply* nimmt nun den kürzesten Weg zu Knoten A (10-12).

2.1.5 Daten einbringen

Die Benutzer stellen im Gegensatz zu den anderen Peer-to-Peer-Diensten nicht einfach ein ganzes Verzeichnis zur Verfügung, sondern müssen ihre Daten explizit ins Freenet einfügen. Die Dateien werden gehashed, um den GUID-Key zu erzeugen, einem SSK zugeordnet (bzw. ein SSK wird neu erzeugt, wenn dieser noch nicht existiert) und schliesslich ins Netz kopiert. Der Knoten verschickt nun ein *Data Insert* (samt Datei) an denjenigen Knoten, der einen Schlüssel besitzt, der dem der neuen Datei am nächsten ist. Dieser speichert lokal eine Kopie der Datei und schickt die *Data Insert* Nachricht an einen nächstpassenden Knoten weiter. Der Vorgang wiederholt sich solange, bis der TTL auf 0 gesunken ist.
Damit erreicht Freenet durch das sofortige Vervielfältigen neuer Daten ein hohes Maß an Redundanz und es kann darüberhinaus kein Zusammenhang zwischen Daten und ihrem eigentlichen Ursprung hergestellt werden.

2.2 Aufbau des Netzes

Aus den im vorherigen Abschnitt genannten Gründen, nennt man Freenet nicht ein File-Sharing sondern ein File-Storage System. Jeder User stellt einfach nur eine bestimmte Menge Speicher zur Verfügung. Neu eingefügte Dateien landen auf beliebigen Knoten und es ist nicht vorhersagbar, wie sich diese Daten im Laufe der Zeit von Knoten zu Knoten bewegen werden.

2.2.1 Neue Knoten

Ein neuer Knoten erzeugt von sich selbst einen Schlüssel, der ihn eindeutig im Netz identifiziert und an eine physikalische Adresse bindet. Den Schlüssel schickt der Knoten in einem sogenannten *New Node Announcement* an einen schon existierenden Knoten, dessen Daten in

irgendeiner Form bekannt sein müssen (z.B. in eine Newsgroup gepostet wurden[4] oder man die Daten von einem Freund bekommt, den man „vertraut“[5]). Dieser trägt den neuen Knoten in seine Routingtabelle ein und schickt das *New Node Announcement* an einen zufälligen gewählten, ihm bekannten Knoten weiter, bis der TTL auf 0 gesunken ist. Dabei handeln die Knoten untereinander noch einen Bereich von Schlüsseln aus, die der neue Knoten initial speichern soll.

2.2.2 Knoten trainieren

Je mehr Dateien der neue Knoten ins Netz einspeist, desto bekannter wird er. Die benachbarten Knoten machen entsprechende Einträge in ihre Routingtabellen und im Gegenzug wird der neue Knoten mehr und mehr Suchanfragen bekommen, auch zu Dateien die er unter Umständen gar nicht besitzt. Sollten diese Anfragen erfolgreich sein, lernt der Knoten neue kennen. Das liegt daran, dass Knoten, die die gesuchten Daten haben, mit einer gewissen Wahrscheinlichkeit ihre eigenen Angaben in *Data Reply* mitsenden. Die eigene Routingtabelle wird dadurch mit immer mehr IP's erweitert und der Benutzer wird im Laufe der Zeit weitere Daten finden können.

2.2.3 Speicherplatzverwaltung

Da Freenet als File-Storage System nur endlich viel Speicher zur Verfügung steht, muss natürlich entschieden werden, welche Dateien als erstes gelöscht werden, wenn der Speicher nicht mehr ausreicht.
Die Strategie, nach der jeder Knoten vorgeht, ist Least Recently Accessed (LRA), äquivalent zu Least Recently Used (LRU). Jeder Knoten führt eine Liste seiner gespeicherten Daten, die nach Popularität geordnet sind. Neu hinzukommende Dateien bzw. wiederholte Anfragen sorgen dafür, dass die Dateien wieder ganz oben eingereiht werden. Daten, nach denen weniger Nachfrage besteht, wandern langsam nach unten. Ist der verfügbare Speicher erschöpft und es kommen weitere Daten hinzu, so werden die untersten Daten als erstes gelöscht.

2.3 Analyse

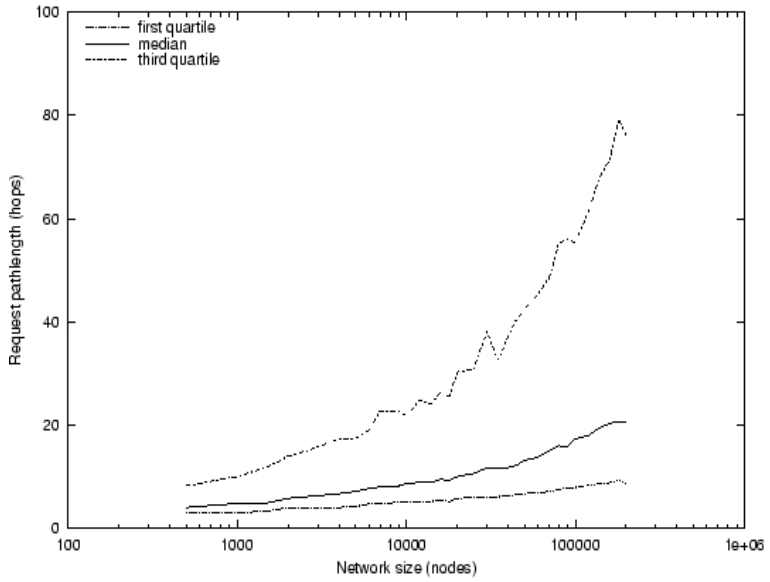
Die folgenden Analysen basieren auf Simulationen [1] und stellen deshalb nur einen bedingten Praxisbezug her. Das mag den Leser enttäuschen, nur sei an dieser Stelle gesagt, dass es nicht möglich ist abzuschätzen, wieviele Knoten es derzeit in Freenet gibt, welche Datenmenge sie halten, und wie redundant manche Dateien enthalten sind. Es ist unmöglich, aussagekräftige, praktische Tests mit dem existierenden Freenet durchzuführen.

2.3.1 Performance

Der Simulationsaufbau sah folgendermaßen aus: Man startete mit 20 initialen Knoten, die in einer Ringstruktur verbunden waren. Nun wurden willkürlige Dateien von zufällig ausgewählten Knoten eingebracht. Nach jeder fünften Datei wurde ein neuer Knoten hinzugenommen, der sein *New Node Announcement* wiederum an einen zufällig gewählten Knoten verschickt hat. Dieser Vorgang wurde solange fortgesetzt, bis das Netzwerk auf 200.000 Knoten angewachsen war. Die folgende Grafik stellt die Pfadlänge für Requests der Größe des Netzwerkes gegenüber.

[4] Man beachte, dass hier ein Schwachpunkt von Freenet liegt. Den Schlüssel existierender Knoten ausserhalb von Freenet bekannt zu geben macht diesen angreifbar.

[5] Das Konzept erinnert an die Kindheit des heutigen Internets. Suchmaschinen gab es noch keine, sondern man konnte andere Seiten nur über Links finden, die die Betreiber auf ihren Seiten angeboten haben.

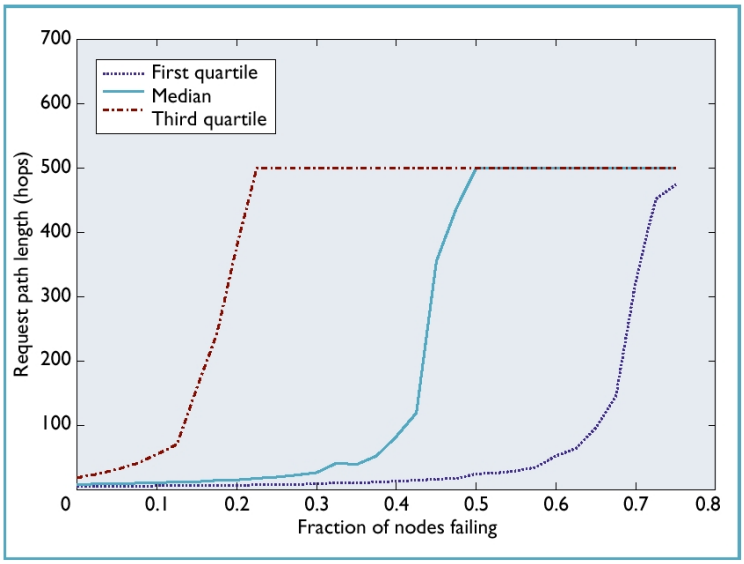


Die Grafik zeigt das Verhältnis der Pfadlänge, also die Anzahl der Hops[6], die nötig sind, um eine Datei zu finden, gegenüber der Anzahl der Knoten. Dabei beschreiben die drei Graphen Dateien in verschiedenen Quartilen.
Wie man sehen kann, verhält sich die Pfadlänge logarithmisch zur Anzahl der Knoten. Eine Hochrechnung auf 1.000.000 Knoten würde ergeben, das nicht mehr als 30 Hops nötig sind, um eine bestimmte Datei zu finden. Ausnahme sind Dateien aus dem dritten Quartil. Dort verhält sich der Graph linear zur Anzahl der Knoten, womit deutlich gezeigt wird, dass Freenet populäre Daten aufgrund der LRA-Strategie der Knoten in den Routingtabellen fördert.

2.3.2 Fehlertoleranz

Aufgrund des redundanten Konzepts, Dateien schon beim Einbringen stark zu vervielfältigen, ist Freenet äusserst fehlertolerant. Analysen haben ergeben, dass selbst bei 30% (zufälligem) Ausfall des Netzes die Performance kaum darunter leidet. Der TTL war noch unter 20, um eine Datei zu finden. Erst bei über 40 % - 45 % Ausfallrate musste man den TTL auf 300 bis 500 Hops erhöhen, um überhaupt noch Dateien zu finden. Die folgende Grafik verdeutlicht dies.

[6]Die Hops geben an, wie oft eine Nachricht Knoten passiert hat. Die Zahl der Hops sollte nicht grösser als der TTL sein, weil sonst eine Nachricht ihr Ziel nicht erreichen würde.



2.3.3 Angriffe

Freenet ist äussert resistent gegen Angriffe jeglicher Art. Sollte z.B. bekannt werden, welcher Benutzer bestimmte Daten anbietet, so hat die Löschung seines Knotens keine negativen Auswirkungen auf den Datenbestand in Freenet, da die Daten beim Einbringen sofort redundant auf mehrere Knoten verteilt werden.
Das Flooden[7] eines Knotens ist nicht möglich, da man die entstehenden Hashwerte für Dateien nicht vorhersagen kann. Ausserdem sind die mit SHA-1 erzeugten Hashwerte dermassen zufällig, dass selbst aus Dateien mit fast identischem Inhalt völlig andere Hashwerte erzeugt werden und somit auch auf unterschiedlichen Knoten gespeichert werden.

3 Zusammenfassung

Freenet wird seinem Ruf, ein absolut anonymes, verteiltes Informationssystem zu sein, gerecht. Es ist nicht möglich, irgendwelche Daten über die Informationsanbieter bzw. -konsumenten herauszufinden. Da ausserdem die Datenkommunikation verschlüsselt erfolgt, macht Abhören keinen Sinn. Dennoch gibt es eine Reihe von Schwierigkeiten, die im Folgenden angesprochen werden sollen.
Ein Schwachpunkt von Freenet ist das Problem des initialen Knoten. Dieser muss sein *New Node Announcement* an einen bekannten Knoten schicken. Die ideale Lösung wäre, dass der Benutzer sein *New Node Announcement* an einen bekannten Knoten schickt, dessen Schlüssel er hat und dem er vertrauen kann. In der Praxis wird dieser Fall wohl eher selten vorkommen und der Benutzer muss sich auf öffentliche, zugänglich gemachte Schlüssel verlassen, die jedoch von bestimmten Interessensgruppen stammen können, die nur ihre Daten fördern und so dem neuen Knoten keine Chance auf Verbreitung seiner Daten lassen.
Freenet ist ein sehr komplexes System. Vom Benutzer wird zur Zeit noch relativ viel „know-how“ verlangt. Viele Menschen werden deshalb eher zu den leichteren, einfacher zu erlernen-

[7] Neudeutsch für „mit Daten überfluten“. Das Angriffsziel wäre hier, soviel Datenverkehr wie möglich auf dem Knoten zu erzeugen, so dass die wichtigen Daten in der Routingtabelle nach unten rutschen und nach und nach vom Knoten gelöscht werden.

den Alternativen greifen.
Weiterhin wurde momentan noch keine Suchfunktion implementiert. Das existierende Konzept lehnt sich zur Zeit noch an die Änfänge des Internets an, auf dem jeder Benutzer auf seiner Seite Links zu weiteren Daten bereitgestellt hat. Genauso stellt in Freenet ein Benutzer weitere Daten seines Subspaces zur Verfügung bzw. die CHKs oder SSKs anderen Leute, die er schon kennt. Nur wird man auf diese Art niemals so viele Daten erfassen, wie dies eine Suchmaschine tun könnte.
Das Problem der fehlenden Suchfunktion steht momentan im Fokus der Entwickler und man darf auf nächste Programmversionen gespannt sein[8].

Literatur

[1] The free network project - liberty through technology, http://www.freenetproject.org

[2] Peer-to-Peer Networking, Protecting Free Expression Online with Freenet, http://computer.org/internet, 01/2002

[3] Ian Clarke, A Distributed Decentralised Information Storage and Retrieval System, Division of Informatics, University of Edinburgh, 1999

[4] http://www.gnutella.com

[5] The Free Haven Project. http://www.freehaven.net

[6] EDonkey 2000, http://www.edonkey2000.com

[7] Kazaa and Grokster, http://www.kazaa.com, http://www.grokster.com

[8] Zur Entstehungszeit dieses Dokumentes war aktuelle Version von Freenet 0.4