Seminar: Peer-To-Peer Overlay Network Systems S15

Freenet: A Distribyted Anonymous Information Storage and Retrieval System

Dennis Hägler
Freie Universität Berlin
dennis.haegler@fu-berlin.de

16. August 2015

Zusammenfassung

Freenet ist ein Peer-To-Peer Overlay Netzwerk und dient für den anonymen Austausch von Dateien. Die Funktionweise von Freenet, die Umsetzung der Anonymität im Freenet, sowie der Aufbau von Freenet werden in dieser Ausarbeitung aufgeführt. Als Vorlage diente das Paper [1].

1 Einleitung

Das Ziel von Freenet besteht den anonymen Austausch von Informationen zu ermöglichen und abzuspeichern. Dieses Ziel soll durch Dezentralisierung, Redundanz, Verschlüsselung und dynamisches Routing erreicht werden. Es wurde als freie Software unter der GNU General Public License entwickelt.

2 Architektur

Ein Freenet-Netzwerk besteht aus Nutzer-Knoten, die ihren lokalen Speicher anderen Nutzern zur Verfügung stellen. Jeder Knoten hat einen direkten Nachbarn, von dem gelesen wird und einen direkten Nachbarn, zu dem Dateien verteilt werden. Des Weiteren verwaltet jeder Knoten eine dynamische Routingtabelle. Jede Routingtabelle beinhaltet die Adressen von Knoten und deren Schlüssel. Genutzt wird Freenet, um die eigene Speicherkapazität zu erhöhen und um ungenutztem Speicher zur Verfügung zu stellen.

Anfragen für Schlüssel von Knoten zu Knoten erfolgen über eine Kette von Proxy-Requests, indem jeder Knoten entscheiden kann, an wen der nächste Request geht. Dieses Verfahren ähnelt dem IP-Routing. Jede Route ändert sich abhängig vom angefragten Schlüssel. Um die Privatsphäre der Knoten zu unterstützen, haben diese nur Informationen über den ummittelbaren Nachbarn (up- und downstream).

Jeder Request erhält einen hops-to-live Wert, der sich bei jedem erreichten Knoten verringert. Dadurch werden Endlosschleifen vermieden. Jeder Request enthält außerdem eine pseudo-unique Zufalls-ID, um Endlossschleifen zu verhindern. Jeder Request, der zum zweiten mal bei einem Knoten ankommt, fliegt raus. Es gibt keine Herarchie der Knoten. Alle Knoten sind gleichgestellt im Freenet.

2.1 Schlüssel

Dateien im Freenet werden mit einem 160 Bit SHA-1 Schlüssel dargestellt. Freenet hat 3 verschiedene Schlüssel, die einen bestimmten Teil einer Datei beschreiben.

2.1.1 Keyword-Signed Schlüssel

Der Keyword-Signed Schlüssel dient für die Beschreibung einer Datei im Freenet. Jeder Benutzer, der eine Datei anlegt, wählt einen Text zur Idenfizierung dieser Datei, wie z.B.

text/philosophy/sun-trz/art-of-war

Aus diesem Text wird deterministisch ein public-private Schlüsselpaar generiert. Der Public Schlüssel wird anschließend gehasht, um den Dateischlüssel zu erhalten, der dann mit dem privaten Schlüssel signiert wird. Der Dateischlüssel ist der Schlüssel, mit der die Datei dargestellt wird. Dieser Schlüssel kann mit dem gewählten Text erneut berechnet und gesucht werden. Um anderen Nutzern den Zugriff auf die Datei zu gewähren, verschickt der Autor der Datei den beschreibenden String. Keyword-Signed Schlüssels sind so leicht zu merken. Ein Problem bei dem Keyword-Signed Schlüssel ist der geringe Namespace, welcher dazu frühren kann, das unterschiedliche Dateien die selbe Bezeichnung und dadurch den selben Dateischlüssel erhalten. Um das Problem mit der Namenskollidierung zu umgehen dient der Signed-Subspace Schlüssel.

2.1.2 Signed-Subspace Schlüssel

Der Signed-Subspace Schlüssel ist eine Erweiterung des Keyword-Signed Schlüssel. Der Schlüsselt erweitert den Namespace und reduziert Kollisionen, wo zwei verschieden Dateien den selben Schlüssel erhalten können. Ein zufällig gewähltes asymmetrisches Schlüsselpaar stellt den Namespace dar. Um eine Datei hinzuzufügen, wählt der Nutzer einen beschreibenden Text. Der public Signed-Subspace Schlüssel und der beschreibende Text werden unabhängig voneinander gehasht, anschließend miteinander geXOR't und dann wieder gehasht.

2.1.3 Content-Hash Schlüssel

Der Content-Hash Schlüssel repräsentiert den Inhalt der Datei in Form eines Hashes. Jede Datei erhält dadurch einen pseudo-einzigartigen Schlüssel. Zusätzlich wird eine Datei mit einem zufällig generierten Schlüssel verschlüsselt. Um Zugriff auf die Datei zu ermöglichen, veröffentlicht der Nutzer den Content-Hash Schlüssel und den dazu gehörigen Schlüssel zur Entschlüsselung.

Eine Datei kann kann mit dem Content-Hash Schlüssel durch den Autor aktualisiert werden. Dazu wird ein neuer Content-Hash Schlüssel erzeugt und unter dem Signed-Subspace Schlüssel abgelegt. Der neue Schlüssel muss sich vom alten unterscheiden. Sollte die neue Datei einen Knoten erreichen, der noch auf die alte Version verweist, ensteht eine Kollision, die dazu führt, dass der Knoten die Signatur der neuen Version überprüft, mit der alten Version validiert und auf die neue Version verweist. Der Signed-Subspace Schlüssel verweist immer auf die aktuellste Version der Datei. Die alte Version kann weiterhin erreicht werden, indem direkt der Content-Hash Schlüssel der älteren Version angefragt wird. Wird die ältere Version über einen bestimmten Zeitraum nicht mehr aufgerufen, kann diese vom Freenet entfernt werden.

Eine weitere Funktion vom Content-Hash Schlüssel ist das Aufteilen einer Datei in mehrere Teildateien, um z.B. große Daten mit weniger Bandbreite über das Netzwerk zu verschicken.

2.2 Datenaustausch

2.2.1 Daten empfangen

Um Daten im Freenet zu empfangen, wählt oder berechnet der Nutzer einen Hash. Dieser Hash wird dann als Request-Nachricht über eine Kette von Requests und mit einer gegebenen hops-to-live-Zahl versendet. Als Start dient der eigene Knoten. Der Knoten, bei dem die Request-Nachricht ankommt, überprüft, ob der gesuchte Hash vorhanden ist. Sollte die Datei gefunden werden, sendet der Knoten die Datei zurück. Sollte der gesuchte Schlüssel nicht vorhanden sein, wird der nächste Knoten aus der Routingtabelle genommen, um die Request-Nachricht weiterzuleiten. Dabei wird die hops-to-live-Zahl um eins verringert. Dieser Vorgang wiederholt sich so lange, bis der hops-to-live-Wert auf null

fällt. Kann ein Knoten einen Request nicht mehr weiterleiten, wird der vorherige Knoten benachrichtigt. Dieser sucht sich einen neuen Knoten aus der Routingtabelle und versendet den Request neu. Sollte der hops-to-live-Wert auf null fallen und es wurde keine Datei mit dem Schlüssel gefunden, wird eine Fehlernachricht bis zum Anfang der Kette weitergeleitet. Wird die Datei gefunden, wird dieser Hash in allen Knoten eingetragen, die Bestandteil der Request-Kette waren.

2.2.2 Daten verteilen

Verteilen von Daten ist ähnlich wie das Empfangen von Daten. Der Nutzer berechnet den Hashwert für die Datei und versendet diese an den eigenen Knoten. Dieser überpüft, ob der Hashwert schon vorhanden ist. Sollte das der Fall sein, kommt es zur Kollision und der Nutzer muss einen neuen Schlüssel berechnen. Sollte der Schlüssel nicht vorhanden sein, leitet der Knoten den Request weiter, bis der hop-to-live-Wert aufgebraucht ist oder die Datei gefunden wurde. Sollte die Datei gefunden werden, kommt es wieder zur Kollision und die Fehlernachricht wird bis zum Initialknoten durchgereicht, der dann wieder einen neuen Schlüssel berechnen muss. Sollte die Datei nicht gefunden werden, wird die Datei in allen Knoten eingefügt, die an der Request-Kette beteiligt waren. Sollte ein Knoten keinen Speicherplatz für die neue Datei verfügbar haben, wird die am wenigsten aufgerufene Datei ersetzt. Freenet garantiert somit keine permanente Persistierung von Daten sicher.

2.2.3 Daten verwalten

Der Speicher eines Knotens wird als LRU (Last Recent Used) cache verwaltet. Die Dateien sind nach letzt genutzt sortiert. Kommt beim Knoten eine neue Datei zum speichern und der Speicher ist voll, wird die am wenigsten genutzte Datei vertrieben, bis wieder Speicherplatz vorhanden ist. Der Eintrag in der Routingtablle bleibt vorerst bestehen. Somit kann eine Kopie der Datei wieder beschafft werden, wenn genug Speicherplatz vorhanden ist. Sollte sich der Knoten mit weiteren neuen Dateien füllen, verschwindet auch der Eintrag in der Routingtabelle der entfernten Datei.

Dateien im Freenet sind nicht permanent verfügbar. Sollten alle Knoten entscheiden die selbe Datei aus ihren Speicher zu vertreiben, ist die im kompletten Netzwerk nicht mehr verfügbar.

Wird eine veraltete Datei weiterhin genutzt, bleibt sie solange im Speicher, wie sie auch abgefragt wird. Jede Datei ist im Freenet verschlüsselt und benötigt einen Schlüssel zum Entschlüsseln. Dieser Schlüssel ist völlig unabhängig von dem Content-Hash-Key. Ohne Schlüssel kennt der Betreiber den Inhalt der Datei nicht und ist somit rechtlich abgeschützt, da jedes Wissen vom Inhalt abgestritten werden kann.

2.3 Knoten hinzufügen

Ein neuer Knoten wählt zufälligen Seed und verschickt diesen als Nachricht über das Freenet-Netzwerk. In dieser Ankündigungsnachricht ist die Adresse und dem Hash des Seeds des neuen Knoten enthalten.

Bei Einkunft einer Ankündigung eines neuen Knotens, generiert der Empfänger einen zufälligen Seed, XOR'ed diesen mit dem empfangenem Hash und hasht das Ergebnis um eine Verbindlichkeit herzustellen. Danach leitet der Empfänger Knoten den neuen Hash zu einem zufällig gewähltem Knoten aus seiner eigenen Routingtabelle. Dieser Prozess wiederholt sich bis der hops-to-live Wert von der Ankündigung des neuen Knoten aufgebraucht ist. Der letzte Knoten, der die Ankündigungsnachricht empfängt, generiert nur einen Seed. Nach Überprüfung der Vereinbarungen in den jeweiligen Knoten, fügt jeder Knoten in der Kette den neuen Knoten in seine Routingtabelle ein.

3 Protokoll Details

Jede Nachricht im Freenet ist Paket-orientiert und beinhaltet selbst erklärende Nachrichten,

sowie eine Transaktions-ID. Adressen von Knoten beinhalten eine Methode für den Transport und einen speziellen Identifikator wie zum Beispiel eine IP und eine Port-Nummer.

3.1 Transaktionen

Eine Transaktion beginnt mit einer Request-Handshake Nachricht von einem Knoten zum anderen. Ist der angefragte Knoten aktiv, antwortet dieser mit einer Reply-Nachricht und spezifiziert die Protokollversion, die der Knoten versteht. Handshakes bleiben mehrere Stunden bestehen, damit der Handshake zwischen den beiden selben beiden Knoten zu einem späteren Zeitpunkt ausgelassen werden kann.

3.2 Nachrichten

Alle Nachrichten im Freenet beinhalten einen zufällig generierten 64-Bit Transaktions-ID, eine hops-to-live Wert und einen Depth-Counter. Zu beachten ist, dass nicht garantiert ist, dass die Transaktions-ID eindeutig ist. Die daraus resultierenden Kollision sind bei einer limitiertem Anzahl von Knoten sehr gering. Der hops-to-live Wert wird immer vom Absender der Nachricht festgelegt und verringert sich bei jeden angekommen Knoten um eins. Der Depth-Counter erhöht um eins bei jedem Knoten.

Um Daten zu empfangen, sendet der zu sendende Knoten eine Request-Data Nachricht mit einer Transaktions-ID, einem hops-to-live Wert, einem Depth-Counter und Schlüssel der Datei nach der gesucht wird. Der Empfängerknoten überprüft den Speicher nach der gesuchten Datei. Sollte die Datei nicht im Speicher des Knoten liegen, wird die Anfrage an einem Knoten aus der Routingtabelle weitergeleitet.

Ist ein Request nach einer Datei erfolgreich, antwortet der angefragte Knoten mit einer Send-Data Nachricht. Beinhaltet in dieser ist die angefragte Datei und die Adresse des Knotens, welches diese Datei auslieferte. Ist ein Request nach einer Datei erfolglos und der hops-to-live Wert aufgebraucht, antwortet der entfernte Knoten mit Reply-NotFound.

Um Dateien in Freenet einzufügen, sendet der Senderknoten eine Request-Insert Nachricht mit einer definierten Transaktions-ID, einem hop-to-live Wert, einem Depth-Counter und einem vorgeschlagen Schlüssel für die Datei. Der entfernte Knoten überprüft seinen Speicher nach dem gesuchtem Schlüssel. Sollte der Schlüssel nicht gefunden werden im Speicher, wird die Nachricht weitergeleitet. Führt die Request-Insert Nachricht zu einer Kollision, da es der vorgeschlagenen Schlüssel schon vergeben ist, sendet der entfernte Knoten entweder eine Send-Data Nachricht, mit der existierenden Datei als Inhalt, oder eine Reply-NotFound Nachricht. Die Reply-NotFound Nachricht wird gesendet, wenn die Datei nicht gefunden wurde, dennoch ein Eintrag in der Routingtabelle auf die Datei referenziert. Sollte es bei einer Nachricht zum Einfügen einer Datei zu keiner Kollision kommen und es sind keine weiteren Knoten vorhanden, bei einen hop-to-live Wert größer Null, antwortet der entfernte Knoten mit einer Request-Continue Nachricht. Diese Nachricht soll dazu dienen den Sender mitzuteilen, dass nicht die Anzahl der Knoten abgefragt werden konnte wie gewünscht. Sollte das Einfügen ohne Kollision enden, erhählt der Initialknoten eine Reply-Insert Nachricht, die dem Knoten mitteilt, dass die Datei unter diesem Schlüssel eingefügt werden kann, dann sendet der Knoten eine Send-Data Nachricht.

4 Sicherheit

Freenet hat das Ziel die Anonymität der Autoren und Leser von Dateien zu schützen. Selbst die Dateien müssen im Freenet vor feindlichen Modifikationen geschützt werden und das Freenet-System muss resistent gegen DOS-Angriffen sein.

Freenet stützt sich auf drei wesentliche Aspekte, die aus der Ausarbeitung von Reiter und Rubin stammen [2]. Der erste Aspekt ist der Punkt der Anonymität. Es soll nicht ermittelt werden können wer eine Nachricht im Netz abgeschickt hat oder an wem diese Nachricht verschickt wurde. Der Aspekt ist die Erkennung

von potentiellen Feinden im Freenet. Zu den potenziellen Angreifern gehören, lokale Lauscher, bösartige Knoten oder eine Zusammenarbeit von bösartigen Knoten. Die dritte Achse ist der Grad der Anonymität.

4.1 Schlüssel

Im Freenet können die angefragten Schlüssel nicht versteckt werden, da sie ein Bestandteil des Routingmechanismuses sind und jede Routing-Tabelle auf die Schlüssel angewiesen ist. Dadurch ist eine Anonymität der Schlüssel nicht gewährleistet. Der Nutzen von Hashes als Schlüssel stellt dennoch eine hohe Unklarheit gegen gelegentlichen Mithörern auf. Gegen Wörterbuchangriffen bleibt sind die Schlüssel dennoch anfällig.

4.2 Anonymität

Die Anonymität von Sender-Knoten in einem Bund aus böswilligen Knoten wird gewährleistet, da ein Knoten der eine Nachricht erhält, nicht bestimmen kann ob der Vorgänger die Nachricht initialisierte oder einfach nur weiterleitete. Durch das zufällige Setzen des Depth-Counters wird das Zurückverfolgen von Requests drastisch erschwert.

Es gibt keinen Schutz gegenüber lokalen Mithörern, die Nachrichten vom Nutzer bis zum ersten Knoten beobachten. Solange der erste Knoten ein mithörender Knoten sein könnte, wird es empfohlen, dass den ersten Knoten für den Einstieg ins Freenet von der eigenen Maschine zu wählen. Nachrichten zwischen Knoten sind verschlüsselt und können nicht mehr abgehört werden. Die einzige Beobachtung bleibt der Ausgang einer Nachricht den Eingang einer Anfrage, was dazu schließen lass kann, das der Knoten der Absender sein könnte.

4.3 Datenquellen

Datenquellen können geschützt werden indem die Felder der Datenquellen regelmäßig zurückgesetzt werden. Im Freenet ist es nicht möglich die Quelle einer Datei zu ermitteln. Liefert der Downstream-Knoten eine Datei, kann diese von dem Knoten selbst stammen oder von einem anderen Knoten, die an den Downstream-Knoten gesendet wurde.

4.4 Daten Manipulation

Daten, die unter den Content-Hash oder den Signed-Subspace Schlüsseln gespeichert werden können nicht manipuliert werden, da unechte Dateien im Freenet nachgewiesen werden können. Die einzige Möglichkeit den Inhalt einer Datei zu manipulieren besteht, wenn der Angreifer eine Kollision verursacht und die Signatur fälschen könnte. Unter den Keyword-Signed Schlüssel gespeicherte Dateien sind anfällig gegen Wörterbuchangriffe.

4.5 Denial Of Service

Weitere Angriffe sind die Denial-Of-Service Attacken. Ein Angreifer könnten versuchen mit einer hohen Anzahl an Abfalldateien das Freenet-Netzwerk zu fluten. Eine möglicher Konter ist es den Nutzer lange Berechnung vor der Einfügung durchzuführen zu lassen. Damit würde das Netzwerk entlastet werden. Ein weiter Methode ist die Aufteilung des Speicher in zwei Segmente. Ein Speicher dient für neue Dateien und ein Speicher für Dateien, die eine gewisse Anzahl an Anfragen aufweisen. Mit dieser Methode würden die neuen Datein nur neue Dateien ersetzen können. Dadurch würden keine Berechnung auf die etablierten Dateien ausgeführt werden. Die Fluten wird dadurch gestoppt, dass wenn ein Knoten eine Anfrage versendet, die vom ersten Knoten schon gestoppt wird, da der Knoten die Datei beinhalten. Eine Verbreitung über das Netzwerk ist dadurch verhindert.

Ein Ziel von Angreifern kann es sein existieren Dateien im Freenet zu ersetzen. Wie bei den DOS-Attacken ist ein Angriff auf die Content-Hash oder Signed-Subspace Schlüssel nicht möglich.

Literatur

[1] Ian Clarke, Oskar Sandberg, B. W. T. W. H. Freenet: A distributed anonymous information storage and retrieval system.

[2] Reiter, R. Anonymous web transactions with crowds. *Communications of the ACM 11*, 3 (1999), 32–38.