Rarest-First- und Choke-Algorithmus sind ausreichend

Björn Kalks
Seminar „Internet Measurment“ ,
Technische Universität Berlin,
FG Betriebs- und Kommunikationssysteme

WS 2008/2009 (Version vom 14. Februar 2009)

Zusammenfassung

Diese Ausarbeitung beschäftigt sich im Rahmen des Seminares „Internet Measurment“ mit dem Paper von Arnaud Legout et al. [2] die das Thema BitTorrent und seine Pieces- und Blocks Auswahlalgorithmen experimentell untersucht haben. Sie kommen dabei zu dem Schluss das die aktuellen Algorithmen von BitTorrent, teilweise mit kleineren Verbesserungen, vollkommen ausreichend sind.

1 Einleitung

Das P2P-Netzwerk BitTorrent erfreut sich immer größerer Beliebtheit und ist eines der größten seiner Art. Schon im Jahre 2006 geht eine Studie von ipoque davon aus dass der „P2P-Traffic [...] je nach Tageszeit [...] 30 bis 70 Prozent des Gesamt-Traffics“ [1] ausmacht und die Kurve zeigt weiter nach oben.
BitTorrent stellt also ein interessantes Gebiet für Studien dar, wobei uns im folgenden vor allem eine experimentelle Untersuchung der beiden Hauptalgorithmen, den Rarest-First- und Choke-Algorithmus, von BitTorrent interessiert. Dabei wird von einem realistischen Versuchsaufbau ausgegangen, der wie beim reelen Einsatz von BitTorrent kein globales Wissen über das Netzwerk bietet. Stellen die beiden vorgestellten Algorithmen nun ein ausreichendes Rüstzeug für die gute Verfügbarkeit von Daten innerhalb des Netzwerkes und eine ausreichende Fairness zwischen den einzelnen Nutzern sicher? Im folgenden wird nun gezeigt dass dem tatsächlich so ist und die beiden Algorithmen dafür ausreichend sind.
Hierfür betrachten wir zuerst den grundlegenden Aufbau von BitTorrent und dessen Funktionsweise und Begriffe. Anschliessend werden die Ergebnisse der jeweiligen Versuche für beide Algorithmen analysiert und bewertet.

2 Hintergrund

In diesem Abschnitt soll ein grober Überblick über die Begriffe und Funktionsweise von BitTorrent gegeben werden, um vorab schonmal einen groben Überlick über die Materie zu gewähren.

2.1 Begriffe

Zu Beginn erstmal einen kurzen Überblick über zentrale Begriffe von BitTorrent:

Pieces und Blocks Dateien werden bei BitTorrent in sogenannte Pieces zerlegt, welche wiederum aus Blocks bestehen. Zu beachten ist dabei das nur vollständige Pieces wiederum zum Download bereitgestellt werden können.

Peer Set Ein Peer Set bezeichnet eine Menge an Peers die ein bestimmt lokaler Peer kennt. Peers in diesem Peer Set, außer dem lokalen Peer, bezeichnet man auch als entfernte Peers.

Interested Hat ein Peer ein Piece welches ein zweiter Peer noch nicht besitzt, so ist der zweite Peer interessiert am ersten Peer.

Choked und Unchoked Im Zustand Choked ist ein Peer von einem anderen Peer wenn dieser ihm keine Daten senden will. Dem gegenüber steht der Zustand Unchoked wenn ein Peer einem anderen Peer Daten senden will.

2.2 BitTorrent

Bei BitTorrent handelt es sich um ein Protokoll für ein P2P-Netzwerk bei dem die Anwendung nicht nur als Client dient, sondern gleichzeitig auch als Server. Einzige Ausnahme beim dezentralen Konzept des P2P-Protokolls bilden hierbei die Tracker, die Informationen zu den Torrents bereithalten, jedoch nicht am eigentlichen Datenaustausch beteiligt sind (vgl. [5] Abschnitt 8). „Alle logistischen Probleme eines Dateidownloads werden durch Interaktionen zwischen den Peers geregelt“ (vgl. [4] Abschnitt 2.2). Wichtiger Bestandteil von BitTorrent sind außerdem die Torrent-Dateien. Hierbei handelt es sich um Dateien die man sich herunterladen kann und die wichtige Informationen wie die Tracker-Adresse, die Piece-Größe und einen Hashwert für jedes einzelne Piece enthält. Wenn man eine Datei herunterladen will, stellt man als neuer Peer eine Anfrage an den Tracker um ein initiales Peer Set zu erhalten. Von nun an meldet sich der Peer alle 30 Minuten oder wenn er sich beendet beim Tracker und übermittelt seinen aktuellen Status und andere Information. Fällt die Anzahl der Peers in einem Peer Set unter einen bestimmten Wert, normalerweise 20, so wird der Tracker erneut nach Peers befragt. Ein Torrent ist also grob gesagt eine Ansammlung von Peers die sich gegenseitig kennen und miteinander kommunizieren (vgl. [5] Abschnitt 10) um Daten auszutauschen. Die eigentliche Logik hinter diesem Datenaustausch leisten dann der Rarest-First- und Choke-Algorithmus.

2.3 Piece- und Peer-Auswahlverfahren

Im folgenden Abschnitt werden nun kurz die beiden, von BitTorrent verwendeten, Piece- und Peer Auswahlverfahren in ihrer Funktionsweise erläutert, die, wie wir später sehen werden, zentrales Bestandteil der guten Verteilung von Daten innerhalb von BitTorrent sind.

2.3.1 Rarest-First-Algorithmus

Um möglichst effizient zu sein, nutzt BitTorrent den sogenannten Rarest-First-Algorithmus als Auswahlstrategie für Pieces.
Beim Rarest-First-Algorithmus wird in jedem Peer-Set die aktuelle Anzahl der Kopien eines Piece gespeichert. Anhand dieser Information werden dann bei der Auswahl eines Pieces, dasjenige Piece genommen welches die wenigsten Kopien im Peer-Set hat.
Um bestimmte negative Effekte vom Rarest-First-Algorithmus zu vermeiden wird er normalerweise durch folgende drei zusätzlichen Varianten verbessert:

Random-First-Policy Die ersten vier Pieces werden zufällig ausgewählt um diese Pieces schneller zu erhalten als mit dem Rarest-First-Algorithmus.

Strict-Priority-Policy Wenn man ein Piece eines Blocks nachgefragt hat, werden die restlichen Blöcke mit höchster Priorität nachgefragt um möglichst schnell komplette Pieces zu erhalten.

End-Game-Mode Dieser Modus startet wenn der Download fast komplett ist, wobei in der Spezifikation von BitTorrent keine feste Grenze existiert bei der dieser Modus startet. Im End-Game-Mode werden im gesamten Peer Set alle noch fehlenden Blocks nachgefragt und diese Nachfrage erst mit Erhalt des jeweiligen Blocks abgebrochen. Hierdurch soll eine Verlangsamung des Herunterladens gegen Ende des gesamten Downloads verhindert werden, wenn es passieren kann dass man nur noch langsame Peers zur Verfügung hat, die die letzten Pieces bereitstellen. Dieses Phänomen ist auch als Last-Pieces-Problem bekannt. Für nähere Informationen hierzu sei der Leser auf [3] verwiesen.

Später wird gezeigt das dieser relativ einfache Rarest-First-Algorithmus vollkommen ausreicht um eine möglichste große und schnelle Diversifikation von Daten innerhalb des P2P-Netzwerkes zu gewährleisten.

2.3.2 Choke Algorithmus

Um eine möglichst effiziente Verbreitung von Kopien eines Torrents zu erreichen, wird der Choke-Algorithmus verwendet, der bestimmt wann Daten zu einem entfernten Peer geschickt werden (Unchoked) und wann nicht (Choked). Alle entfernten Peers die im Zustand Unchoked sind und Interesse an dem lokalen Peer haben, befinden sich dabei im sogenannten aktiven Peer Set des lokalen Peers.
Grundsätzlich unterscheidet der Algorithmus dabei ob man gerade Leecher oder Seed ist und verfährt jeweils nach einem bestimmten Schema:

Leecher Alle 10 Sekunden werden die drei schnellsten Peers anhand ihrer Download-Rate ausgewählt und in den Zustand Unchoked versetzt und alle 30 Sekunden wird außerdem ein zufälliger Peer in diesen Zustand versetzt. Hierbei spricht man dann von einem Optimistic-Unchoke (vgl. [5] Abschnitt 11.5). Dieser dient dazu neue Peers zu entdecken und ihre Geschwindigkeit zu testen bzw. ihnen ein erstes Piece zu geben, um neue Kapazität im Netzwerk zu erschliessen.

Seed Die Peers die im Zustand Unchoked sind bzw. Interesse an einem Piece haben, werden anhand der vergangenen Zeit seit ihrem letzten Unchoke ausgewählt, wobei Peers mit kürzeren Zeitspannen bevorzugt werden. Während der Zeit von zweimal 10 Sekunden werden, ähnlich wie als Leecher, die drei ersten Peers weiterhin im Zustand Unchoked belassen und ein vierter Peer zufällig bestimmt (Optimistic-Unchoke). Diese vier Peers bleiben anschliessend weitere 10 Sekunden im Zustand Unchoked.

Später wird gezeigt das dieser neue Choke-Algorithmus vollkommen ausreicht um eine möglichste große Fairness innerhalb des P2P-Netzwerkes beim Austausch von Daten zu gewährleisten.

2.4 Experimenteller Aufbau

Legout et al. haben einen praktischen Aufbau für ihre Experimente gewählt. Sie haben hierfür 26 Torrents mit unterschiedlichsten Eigenschaften ausgesucht. Es gibt dabei Torrents mit wenig oder gar keinen Seedern, aber vielen Leechern. Auf der anderen Seite aber auch Torrents mit mehr Seedern als Leechern. Man wollte hier einen möglichst großen Querschnitt erreichen (vgl. [2] Abschnitt 3.2). Als Anwendung diente *mainline* [1], „das als Referenz zum BitTorrent Protokoll angesehen wird“ (vgl. [2] Abschnitt 3.1). Anschliessend wurden für diese Torrents 8 Stunden dauernde Testläufe gefahren und mitgeloggt. Für weitere Details des Versuchaufbau's sei der geneigte Leser auf [2] (Abschnitt 3) verwiesen.

3 Auswertung für den Rarest-First-Algorithmus

Das Hauptziel von BitTorrent besteht in der möglichst schnell und verstreuten Verbreitung von Dateien. Dies soll der Rarest-First-Algorithmus sicherstellen. Dieses Kapitel beschäftigt sich nun damit, wie man (mit Hilfe einer Entropie) die Leistungsfähigkeit von verschiedenen Torrents vergleichen kann und diese Leistungsfähigkeit möglichst maximiert.

3.1 Entropie

Da man durch den reelen Aufbau des Experiments kein globales Wissen über alle Torrents besitzt, definieren wir eine Entropie als Verfügbarkeit eines bestimmten

[1] von Bram Cohen entwickelter Orginal-Client von BitTorrent

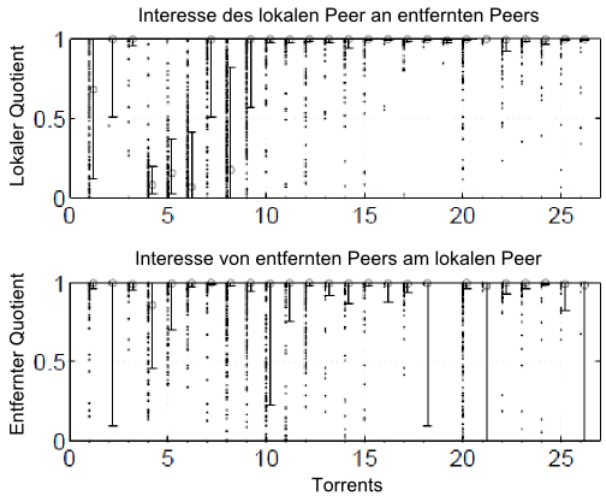

Abbildung 1: Entropien der 26 Torrents des Experiments:
Für jeden entfernten Leecher gibt es einen Punkt, die Kreise stellen den Durchschnittswert und die durchgezogene Linie den Wert von 20% (unten) bis 80% (oben) dar (Quelle: [2] Figure 1).

Pieces. Gezeigt werden soll nun, dass man mit dem Rarest-First-Algorithmus eine nahezu ideale Entropie von 1 erreichen kann, in der jeder Leecher immer an mindestens einem anderen Peer interessiert ist. Als Entropie definieren wir hierzu für jeden entfernten Peer zwei Quotienten:

Lokaler Quotient Stellt die Zeit die ein lokaler Peer als Leecher Interesse an einem entfernten Peer hat in Relation zu der Zeit die dieser entfernten Peer im Peer Set des lokalen Peer's, solange dieser Leecher ist, tatsächlich zugebracht hat.

Entfernter Quotient Stellt die Zeit die ein entfernter Peer als Leecher Interesse an einem entfernten Peer hat in Relation zu der Zeit die dieser entfernten Peer im Peer Set des lokalen Peer's, solange dieser Leecher ist, tatsächlich zugebracht hat.

Betrachtet man nun die beiden Quotienten für die 26 Torrents des Experiments (vgl. Abbildung 1), so sieht man das im Durchschnitt 80% beim lokalen Quotienten und 90% beim entfernten Quotienten nahe an der idealen Entropie von 1 liegen. Ein paar der Torrents beim lokalen Quotienten erreichen dieses Ziel jedoch nicht. Schaut man sich diese speziellen Torrents genauer an, erkennt man das sie sich noch in der kritischen Startphase befinden und noch nicht alle Pieces mindestens einmal vervielfältigt wurden. Die schlechtere Entropie lässt sich nun dadurch erklären, dass Torrents die sich in der Startphase befinden den Flaschenhals darstellen. Sie können ihre seltenen Pieces zu diesem Zeitpunkt nur selbst verteilen, wobei ihre eigene Upladegeschwindigkeit dann die Grenze der Leistungsfähigkeit innerhalb des P2P-Netzwerkes darstellt. Sobald ein Piece jedoch 2 mal vorhanden ist, spielt BitTorrent seine Stärke aus und die Geschwindigkeit des Verteilens steigt exponentiell an, da jetzt jedes ganze verteilte Piece wieder dem Netzwerk zur Verfügung steht und von anderen Peers heruntergeladen werden kann.

Auch beim entfernten Quotienten gibt es solche Ausreißer. Der Durchschnitt der Entropie ist (außer beim Torrent 4 und 19 [2]) zwar immer nahe 1, es gibt jedoch bei einigen Torrents eine gewisse Prozentzahl von Peers die hiervon stark abweichen, wie zum Beispiel Torrent 3 und 18. Legout et al. [2] führen dies auf 2 Hauptgründe zurück: Erstens gibt es Peers die schon alle Pieces haben wenn sie dem Peer Set beitreten und zweitens gibt es Peers die sich im sogenannten Super-Seeding-Modus befinden und kein Interesse an Pieces haben. Für nähere Informationen zum Super-Seeding-Modus sei auf [5] verwiesen. Auch wenn dieser Modus nicht zum eigentlichen BitTorrent-Standard gehört, ist er dennoch häufiger anzutrefen.
In allen der oben genannten Fälle kann der Rarest-First-Algorithmus also keine annähernd ideale Entropie sicherstellen, jedoch rechtfertigt dies nicht die Ersetzung des Algorithmus durch andere Varianten, da diese Fälle relativ selten auftreten bzw. man ja durchschnittlich sehr nahe am Ideal ist und die ggf. minimalen Verbesserungen keine komplexeren Algorithmen rechtfertigen.
Wir stellen also fest das der Rarest-First-Algorithmus eine nahezu ideale Entropie sicherstellt.

3.2 Analyse des Rarest-First-Algorithmus

Bei der Analyse des Rarest-First-Algorithmus kann man zwei verschiedene Zustände unterschieden. Als erstes einen Übergangszustand (transient state) in dem es nur einen Seed gibt und somit noch seltene Pieces existieren die nur der Seed besitzt. Dieser Zustand ähnelt dem des Initial Seed. Will man eine möglichst rasche Verteilung eines Torrents erreichen, sollte man bestrebt sein (zum Beispiel durch den Rarest-First-Algorithmus) die Zeit die ein Torrent in diesem Zustand verbringt zu minimieren.
Befindet sich ein Torrent nicht im Übergangszustand, so ist er im Beständigen-Zustand (steady state) und es gibt keine raren Pieces mehr. Da man in diesem Zustand eine exponentielle Verteilung erreichen kann im Gegensatz zur linearen Verteilung im Übergangszustand, sollte der Algorithmus verhindern das man wieder in den Übergangszustand abrutscht.

3.2.1 Analyse des Übergangszustands

Solange sich ein Torrent im Übergangszustand befindet, ist die Uploadkapazität des initialen Seeds selbst die obere Grenze bei der Verbreitung, da nur er seltene Pieces besitzt und sie verteilen kann. Da der Rarest-First-Algorithmus seltene Pieces zuerst vervielfältigt nutzt er diese Uploadgeschwindigkeit voll aus und minimiert so die Zeit im Übergangsszustand. Wurde ein Piece mindestens einmal so vervielfältigt, steigt die Kapazität im Peer Set für dieses Piece exponentiell. Sobald dies dann auf alle Pieces zutrifft verlässt der Torrent den Übergangszustand und wechselt in den Beständigen-Zustand den wir als nächstes betrachten werden.

[2] dieser Torrent hat gar keine Auswertung des Quotienten auf Grunde der sehr geringen Anzahl an Leechern

3.2.2 Analyse des Beständigen-Zustands

Um zu untersuchen ob ein Torrent vom Beständigen-Zustand wieder in den Übergangszustand zurück fallen kann, muss man sich anschauen ob es passieren kann dass aus einem Piece von dem es mindestens zwei Kopien gibt, wieder ein seltenes Piece werden kann. Theoretisch ist dies möglich wenn Peers das Peer Set verlassen und so bestimmte Pieces nicht mehr zur Verfügung stehen. Der Rarest-First-Algorithmus bekämpft jedoch diese Möglichkeit bestmöglich durch sein Verhalten. Sobald ein Peer das Peer Set verlässt werden die Anzahl der Pieces im Peer Set neu berechnet. Falls durch das Verlassen Pieces zu seltensten [3] Pieces werden, werden diese automatisch wieder schnellstmöglich durch Algorithmus vervielfältigt. Der Rarest-First-Algorithmus hält hier eine möglichst konstante Anzahl an Pieces in einem Peer Set. Legout et al. [2] haben hierzu in ihren Versuchen keinen einzigen Torrent gefunden der vom Beständigen-Zustand in den Übergangszustand zurückfiel.

Es konnte also gezeigt werden dass der Rarest-First-Algorithmus die Zeit im Übergangszustand minimiert und verhindert dass man vom Beständigen-Zustand wieder zum Übergangszustand zurückkehrt.

4 Auswertung für den Choke-Algorithmus

Beim Choke-Algorithmus handelt es sich um ein Verfahren zur Auswahl von Peer zu denen Daten gesendet werden. Im folgenden betrachten wir vor allem den Fairness-Aspekt dieses speziellen Algorithmus, „da der Choke-Algorithmus unbestritten eine effiziente Peer-Auswahlstragie ist“ (vgl. [2] Kapitel 4.2). Zuerst schauen wir uns dazu die Fairness allgemein an und anschliessend jeweils als Leecher bzw. Seed.

4.1 Fairness des Algorithmus

Will man über die Fairness von P2P-Netzwerken sprechen, muss man erst einmal festlegen welche Beteiligten es in solch einem Netzwerk gibt und welche Beziehungen es zwischen ihnen gibt.
In einem P2P-Netzwerk kann man drei verschiedene Akteure unterscheiden: Seeds, Leecher und Free-Riders, also Akteure die selbst keine Daten im Netzwerk zur Verfügung stellen. Ein Hauptkritikpunkt an der Fairness des Choke-Algorithmus liegt darin das er nicht auf dem Prinzip des *Geben und Nehmens* beruht. Dies würde bedeuten das wenn jemand k Bytes heruntergeladen hat, er auch wiederum k Bytes selbst zum Herrunterladen bereithalten müsste. Hiermit würe man Free-Rider bestrafen, da sie dem Netzwerk zwar Ressourcen entziehen jedoch keinen Nutzen haben. Diese an sich scheinbar sehr faire Variante ist jedoch in der Praxis so gut wie nicht umsetzbar, da zum Beispiel ein Seed nicht die Vervielfältigungsrate eines Leechers bestimmen kann, da der Seed selbst ja keine Daten mehr benötigt und so ohne

[3] Hierbei ist zu beachten dass nicht unbedingt seltene Pieces entstehen in dem Sinne dass das Piece nur noch einmal vorhanden ist.

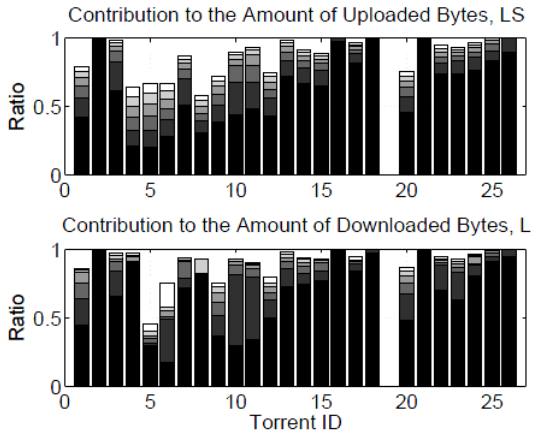

Abbildung 2: Fairness des Choke-Algorithmus im Leecher-Zustand:
Gleiche Farben in beiden Diagrammen stellen gleiche Peer Sets dar, wobei jedes Peer Set aus 5 entfernten Peers besteht die nach Anzahl der heruntergeladenen Bytes sortiert sind (Quelle: [2] Figure 7).

globales Wissen keine Chance hat diese Vervielfältigungsrate zu bestimmen. Desweiteren „können Leecher eine asymmetrische Netzwerkverbindung haben“ (vgl. [2] Kapitel 4.2.1). Dies bedeutet das die Upload-Rate ungleich der Download-Rate ist. In diesem Falle würde der Leecher in einem strikten *Geben und Nehmen* Szenario nie seine Downloadkapazität ausnutzen können.
Aus diesen Gründen haben sich Legout et al. [2] für folgende zwei Fairnesskriterien (vgl. [2] Abschnitt 4.2.1) entschieden:

- Leecher die eine höhere Uploadgeschwindigkeit haben als andere Leecher, bekommen eine höhere Downloadgeschwindigkeit.
- Jeder Seed teilt jedem Leecher die selbe Download-Zeit zu.

Mit diesen beiden Kriterien ist es möglich dass Free-Rider überflüssige Kapazität nutzen können, aber Leecher die selbst Kapazität zur Verfügung stellen bevorzugt werden bei der Geschwindigkeit. Seeds unterscheiden nicht zwischen Leechern die Kapazität bereitstellen und Free Ridern, jedoch beeinträchtigt dies nur geringfügig die Fairness des Systems, da Free-Rider laut dem ersten Kriterium immer weniger Geschwindigkeit beanspruchen können als Leecher die auch bereitstellen. Es bleibt zu zeigen das Kriterium für Leecher eine Vervielfältigung fördert und dass das Kriterium für Seeds vom Choke-Algorithmus erfüllt wird.

4.2 Verhalten als Leecher

Um zu zeigen dass das Kriterium für Leecher eine Vervielfältigung fördert, betrachten wir Abbildung 2. In dieser Abbildung symbolisieren gleiche Farben in beiden Diagrammen jeweils gleich Peer Sets. Bis auf die Ausnahme Torrent 19 und 5, zu denen wir etwas später genauer kommen, zeigen alle Torrents das gewünschte Verhalten, da die beiden Graphen für einen Torrent in beiden Diagrammen sehr ähnlich

sind. Es besteht eine starke Beziehung zwischen der Menge an hoch- und heruntergeladenen Bytes (vgl. [2] Kapitel 4.2.2). Torrent 19 taucht in den beiden Diagrammen gar nicht als Graph auf, da dieser Torrent auf Grunde der geringen Anzahl an Leechern nie Daten hoch geladen hat. Torrent 5 hingegen zeigt ein nicht gewolltes Verhalten, da er sich jedoch in der Startphase befindet, lässt sich diese Anomalie dadurch erklären, dass der lokale Peer zwar Daten von einem entfernten Peer heruntergeladen hat die er wollte, dieser jedoch kein Interesse an Daten des lokalen Peers hatte. Dies lässt sich u.a. durch die geringe Entropie in der Startphase erklären.

Nicht direkt eine Beitrag zur Fairness, jedoch zur allgmeinen Funktion des P2P-Netzwerkes trägt der Choke-Algorithmus mit dem Optimistic-Unchoke bei. Während der Choke-Algorithmus ohne Optimistic-Unchoke nur darauf bedacht ist die Verbindung zu den drei schnellsten Peers bestehen zulassen, braucht man aber den Optimistic-Unchoke zum Entdecken / Testen von neuen Peers bzw. um das Netzwerk an sich am Laufen zu halten, indem man auch neuen Peers die noch gar keine eigenen Daten zum verteilen haben, eine Chance gibt ihr erstes oder auch zweites Piece zu bekommen (vgl. [2] Abschnitt 4.2.2).
Trotz alledem konnte man zeigen, dass das Kriterium für Leecher die gewollte Förderung von Vervielfältigung bereitstellt.

4.3 Verhalten als Seed

Legout et al. [2] haben in ihren Versuch festgestellt das „der neue Choke-Algorithmus im Seed-Zustand der Einzige ist der allen Leechern die gleiche Download-Zeit zur Verfügung stellt“ (vgl. [2] Kapitel 4.2.3). Sie haben hierfür drei Gründe ausgemacht:

- Jeder Leecher bekommt die gleiche (kurze) Download-Zeit, was direkt die Diversität der Pieces erhöht, im Gegensatz zum alten Algorithmus in der es möglich war das ein Leecher, ohne das Kriterium, alle Pieces erhalten konnte, was die Diversität verschlechtert hat.
- Free-Rider können nicht mehr herunterladen als jemand der Upload bereitstellt. Im alten Algorithmus konnte sich ein Free-Rider mit hoher Download-Rate den ganzen Torrent herunterladen ohne dem Netzwerk etwas zurückzugeben.
- Der Übergangszustand (siehe auch Rarest-First-Algorithmus) wird verbessert, da die Diversität von Pieces gefördert wird, indem sie auf möglichst viele Peers verteilt werden. Im alten Algorithmus konnte es passieren, dass ein Initial Seed seine raren Pieces einem einzige Peer geschickt hat und dieser anschliessen gegangen ist und der Torrent wieder nur ein Initial Seed war.

Man kann also festellen das der neue Choke-Algorithmus jedem Leecher die gleiche Download-Zeit zur Verfügung stellt und eine Verbesserung gegenüber der alten Variante ist.

5 Zusammenfassung

Die getätigten Versuche mit BitTorrent in einem realistischen Versuchsaufbau konnten tatsächlich zeigen das für ein effizientes P2P-Netzwerk im Stile von BitTorrent die einfachen Rarest-First- und Choke-Algorithmen ausreichend sind und es keiner Ersetzung durch komplexere Algorithmen bedarf.
Speziell sorgt der Rarest-First-Algorithmus darfür das eine nahezu ideale Entropie und eine gute Diversität erreicht wird und das Last-Piece-Problem mit Hilfe des End-Game-Modus gemildert wird. Torrents in der Startphase mit einer geringen Entropie hängen nur von der Kapazität des Initial Peers ab und diese lässt sich nicht mehr verbessern.
Der Choke-Algorithmus erreicht mit den beiden Kriterien eine gute Fairness und macht ein striktes Prinzip des *Geben und Nehmens* überflüssig. Desweitern fördert der neue Choke-Algorithmus die Vervielfältigung und wird nicht durch Free-Rider gestört.
Es bestehen zwar durchaus noch Gebiete innerhalb von BitTorrent die verbessert werden können, jedoch ist die aktuelle Version schon sehr gut geeignet für das schnelle Verteilen von Daten über das P2P-Netzwerk.

Literatur

[1] golem.de: BitTorrent überholt eDonkey, http://www.golem.de/0610/48522.html, Artikel vom 23.10.2006

[2] Arnaud Legout, G. Urvoy-Keller and P. Michiardi: Rarest First and Choke Algorithms Are Enough, Rio de Janeiro, Oktober 2006.

[3] Arnaud Legout, G. Urvoy-Keller and P. Michiardi: Rarest First and Choke Algorithms Are Enough - Technical Report, Rio de Janeiro, Oktober 2006.

[4] Bram Cohen: Incentives Build Robustness in BitTorrent, 22. Mai 2003

[5] Bittorrent Protocol Specification v1.0: http://wiki.theory.org/BitTorrentSpecification, 23. Oktober 2008