BitTorrent

Simon Mittelberger
Betreuer: Benedikt Elser
Seminar Future Internet SS2009
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München
Email: simon.mittelberger@in.tum.de

***Kurzfassung*—Man stelle sich vor man besitze ein Musikstück, ein Video, oder eine beliebige andere Datei, welche relativ groß ist und sehr viele andere Benutzer interessiert. Soll diese Datei nun an viele Benutzer verteilt werden, wird ein Server und genügend Bandbreite benötigt, welche ein normaler Computeranwender nicht zur Verfügung hat. BitTorrent bietet eine Lösung zu diesem Problem.**
BitTorrent ist ein Peer-to-Peer System durch welches Dateien dezentral verteilt werden können. Die Last wird nicht von einem oder einer Gruppe von Servern getragen, sondern wird von jedem einzelnen Client mitgetragen. Ein so genannter Tracker wird benötigt um die Clients zu koordinieren. Er teilt einem Client mit, welcher andere Client Teile einer Datei besitzt. Der Client entscheidet selbständig von welchen Clients er lädt und an welche anderen Clients er verteilt.
Verschiedene Strategien regeln, die Reihenfolge in der die Teile der Dateien von anderen Clients geladen werden und an welche anderen Clients Teile der Dateien verteilt werden.

***Schlüsselworte*—BitTorrent, Torrent, Filesharing, Tit for Tat, Pareto Effizienz, Gefangenendilemma, Peer, Seeder, Leecher, Chunk, Choking-Algorithmus, Peer-to-Peer, P2P**

I. Einleitung

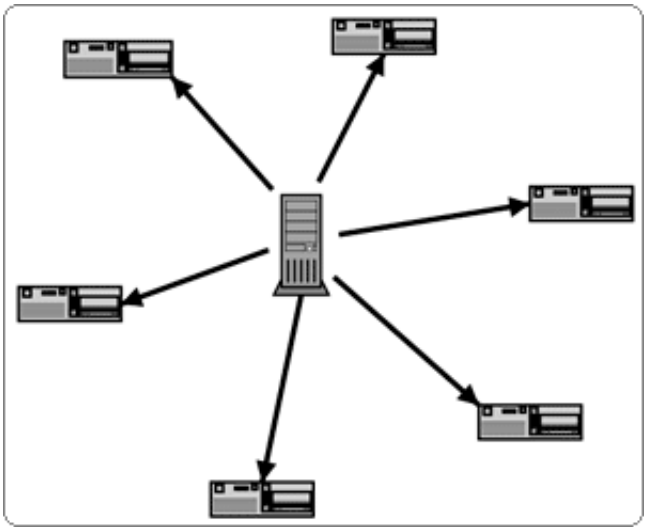

Abbildung 1. Client-Server Modell: Daten-Verteilung auf herkömmlichem Weg; ein zentraler Server verteilt die Daten an viele Clients. [1]

Um eine Datei auf herkömmlichem Weg unter verschiedenen Nutzern des Internets zu verteilen benötigt man einen bzw. mehrere Server, welche die Datei hosten und eine Schnittstelle zum Download für die Clients anbieten. Will man nun eine große Datei unter sehr vielen Benutzern verteilen, benötigt der Server ausreichend Leistung, was heutzutage kein Problem mehr darstellt, und genügend Bandbreite. Die Bandbreite des Servers ist in solchen Szenarien heute meist der Flaschenhals. Der Begriff BitTorrent steht für ein Protokoll zum Datenaustausch über das Internet, für ein Client-Programm, welches dieses Protokoll benutzt, sowie für ein Unternehmen, welches den BitTorrent Client und verschiedene andere Produkte rund um das Thema Datenaustausch über das Internet anbietet.
Das Protokoll BitTorrent wurde im April 2001 von Bram Cohen entwickelt, eine erste Implementation folgte im Juli 2001. Im Jahre 2004 wurde das Unternehmen BitTorrent Inc. von Bram Cohen und Ashwin Navin gegründet, welches auch die Weiterentwicklung des Protokolls vorantreibt.

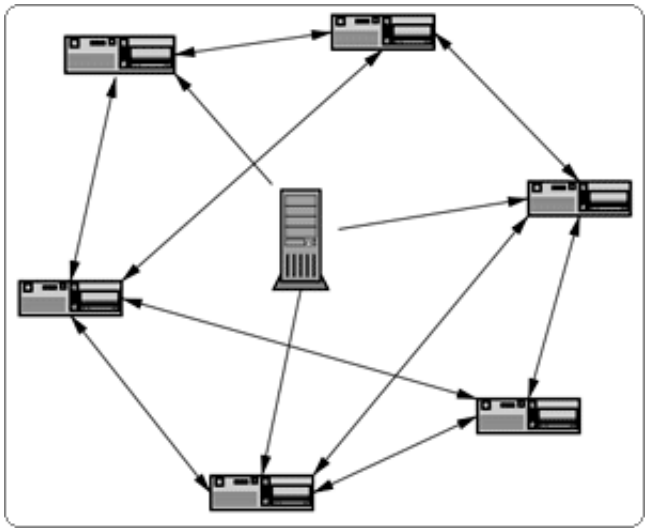

Abbildung 2. Torrent Netzwerk: Ein zentraler Server übernimmt Torrent-Verteilung und Tracking-Aufgabe. Die Clients laden die Datei vom Server und von anderen Clients, was eine Entlastung des Servers zur Folge hat. [1]

Das Protokoll ermöglicht einen Datenaustausch ohne zentralen Server. Jeder Client ist zugleich Uploader und Downloader. Er verteilt die Teile der Datei, welche er bereits empfangen hat an andere Clients weiter, wodurch die Last auf die Clients

aufgeteilt wird.
Über das Torrent Netzwerk lassen sich beliebige Daten übertragen. Man findet von der Linux Distribution und verschiedener Software über Ebooks und Musik bis hin zu Filmen, nahezu alle Arten von Dateien. Vor allem wegen Letzteren hat die Medienindustrie dem Torrent Netzwerk den Kampf angesagt. Torrent Verteiler, wie die schwedische Webseite ThePirateBay.org stehen ganz oben auf der Abschussliste von Warner Bros. Entertainment und Co.
Diese Arbeit beschäftigt sich mit dem Technischen Hintergrund des BitTorrent Systems. Kapitel II befasst sich mit der Client Seite des Systems, Kapitel III mit der System Seite. In Kapitel IV und V wird ein genauerer Einblick in verschiedene Strategien hinter dem BitTorrent System gegeben. Um das Lesen und das Verständnis dieser Arbeit zu erleichtern befindet sich auf der letzten Seite eine Begriffserklärung.

II. Client

Der Download einer Datei über ein BitTorrent Netzwerk gestaltet sich aus der Sicht eines Benutzers als sehr einfach. Der Benutzer lädt eine Torrent Datei über seinen Webbrowser herunter, oder speichert diese über einen anderen Weg auf seinem Computer. Auf den Inhalt der Torrent Datei wird in Kapitel II-B kurz eingegangen. Diese Torrent Datei wird mit dem Client-Programm geöffnet, der Speicherort der Datei wird ausgewählt und schon beginnt der Download. Im Fenster werden nun verschiedene Daten angezeigt, wie zum Beispiel die Download- und Uploadraten, der Fortschritt des Downloads und die Anzahl der Peers mit denen aktuell eine Verbindung unterhalten wird. Diese Einfachheit hat sicherlich wesentlich zur Popularität von BitTorrent beigetragen, vielleicht sogar mehr als seine Performance- und Lastverteilungseigenschaften [2].
2006 liefen über 70 Prozent des deutschen Internetverkehrs über P2P-Systeme. BitTorrent hat die bis dahin bekannteste Tauschbörse eDonkey überholt und verursacht mehr als die Hälfte des gesamten P2P-Verkehrs in Deutschland. Die am häufigsten getauschten Daten seien nach den Aussagen der ipoque GmbH aktuelle Kinofilme, Musik, Serien und Computerspiele. Allerdings ist auch bei Büchern und Hörbüchern ein Zuwachs zu verzeichnen. Derzeit weist BitTorrent, laut BitTorrent Inc., über 160 Millionen installierte Clients weltweit auf [3], [4].
Es gibt eine ganze Reihe von Client-Programmen, die das BitTorrent Protokoll implementieren. Der erste Client wurde von Bram Cohen im Oktober 2002 unter dem Namen BitTorrent herausgegeben. Mittlerweile wird der Client vom Unternehemn BitTorrent Inc. weiterentwickelt und ist seit Version 6.0 keine freie Software mehr. In vielen Clients wurde das BitTorrent Protokoll erst im Nachhinein implementiert, da sie nicht von Anfang an dafür vorgesehen waren. Folgende Liste stellt eine (sehr) kleine Auswahl aller Clients dar:

A. BitTorrent

B. Gnome BitTorrent

C. KTorrent

D. LimeWire

E. qBitTorrent

F. rTorrent

G. Torrent

[5]

III. Technischer Hintergrund

BitTorrent ist ein Peer to Peer System zum Dateiaustausch. Es besteht aus zwei Programmen, dem Client-Programm und dem Server-Programm, genannt Tracker.

A. Peer to Peer (P2P)

Peer to Peer Systeme klassifizieren sich dadurch, dass Teilnehmer, so genannte Peers, sowohl Dienste anbieten, als auch Dienste anderer Teilehmer nutzen. Einfache P2P Netze haben keinen zentralen Punkt, sondern die Peers stellen sich gegenseitig Betriebsmittel und Ressourcen zur Verfügung. Die Weiterentwicklung von P2P Netzen mit zentralen Komponenten nennt man Super Peer Netze. In einer solchen Konfiguration übernehmen so genannte Super Peers die Organisation des Netzes. Das BitTorrent System besitzt in der Regel eine zentrale Einheit, den Tracker, und fällt damit unter die Super Peer Netze. Es gibt aber auch die Möglichkeit des trackerlosen Betriebs, wie später beschrieben wird.

B. Die Veröffentlichung einer Datei über BitTorrent an einem Beispiel

Um zum Beispiel die Datei Ebook.pdf über das BitTorrent Netzwerk zu verteilen, wird eine Torrent Datei, zum Beispiel Ebook.torrent erstellt. Diese Torrent Datei ist sehr klein und enthält Informationen über Ebook.pdf, wie den Namen, die Länge, die Hashwerte der Chunks (Chunks werden später erleutert) und die URL des Trackers. Sie wird über das Internet verfügbar gemacht, zum Beispiel durch einen Webserver, FTPServer, ... Ein Client, der die vollständige Ebook.pdf Datei besitzt muss gestartet werden. Öffnet nun ein Client-Programm die Torrent Datei, kann es die veröffentlichte Datei laden.

C. BitTorrent System

Um eine bessere Performance zu erreichen verteilt das BitTorrent System nicht komplette Dateien, sondern kleinere Einheiten. Diese kleineren Einheiten sind typischerweise 256 KByte große Stücke der Dateien. Man nennt sie Chunks.

1) Der Client: Ein Client lädt Chunk für Chunk einer Datei herunter und setzt diese dann wieder zusammen. Zum Zeitpukt, an dem ein Client einen Chunk fertig geladen hat verifiziert er diesen mit der Prüfsumme aus der Torrent Datei. Nach der Prüfung teilt er dem Tracker mit, dass er im Besitz des Chunks ist. Durch diese Vorgehensweise können die empfangenen Chunks sofort an andere Clients weiterverteilt werden, ohne darauf zu warten, dass die komplette Datei fertig geladen ist. Für die Clients gibt es verschiedene Bezeichnungen:

Sworm	Eine Gruppe von Clients, welche an der selben Datei interessiert sind.
Seeder	Ein Client, der im Besitz aller Chunks einer Datei ist. Er lädt nicht mehr herunter, sondern verteilt nur mehr weiter.
Leecher	Zu diesem Begriff gibt es verschiedene Definitionen. Manche Quellen geben an es handle sich um einen Client, der nur herunterlädt und nicht weiterverteilt [6], andere geben an es sei dasselbe wie ein Peer [7].
Peer	Ein Client, der sowohl herunterlädt, als auch weiterverteilt.

Tabelle I
VERSCHIEDENE BEZEICHNUNGEN FÜR DIE CLIENTS.

2) Der Tracker: Der Tracker verwaltet eine Liste der Clients und ihrer Chunks. Ein Client fordert vom Tracker eine Liste mit Clients an, die einen bestimmten Chunk besitzen. Der Tracker gibt typischerweise eine Liste mit fünf zufälligen Clients, welche im Besitz dieses Chunks sind, zurück. Die Kommunikation mit dem Tracker erfolgt über ein einfaches Protokoll, welches auf HTTP aufbaut. Der Download des Chunks wird zwischen den Clients ausgehandelt und muss nicht immer stattfinden. In Kapitel V: Der Choking Algorithmus wird darauf genauer eingegangen.
Das BitTorrent-Netzwerk existiert nicht als ein gemeinsames Gesamtnetz, wie zum Beispiel eDonkey, sondern vielmehr baut jeder Tracker mit den sich beteiligenden Clients ein eigenes Netz auf. Ein Tracker- bzw. Torrent Anbieter kann sich somit leichter von illegalen Inhalten distanzieren. Ein Tracker kann auch mehrere Dateien verwalten.

D. Trackerloser Betrieb

Der Tracker muss ständig erreichbar sein. Ist er nicht mehr erreichbar können sich die Clients untereinander nicht finden und keine neuen Chunks mehr anfordern. Seit der, im November 2005 erschienen Version 4.2.0 unterstützt der BitTorrent Client den so genannten trackerlosen Betrieb. Die Trackerfunktion wird dabei von den Clients übernommen und ähnlich wie beim Kademilla-Netzwerk als verteilte Hashtabelle auf der Clientseite abgelegt.

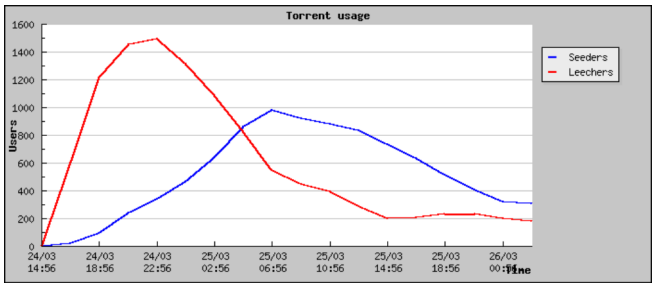

Abbildung 3. Der Graph zeigt die Entwicklung der Seeder und Leecher einer großen Datei (über 400 MByte) über die Zeit. Die Anzahl der Leecher steigt zu Beginn rapide an, erreicht ihren Höhepunkt und fällt dann exponentiell. Die Anzahl der Seeder steigt wesentlich langsamer, erreicht ihren Höhepunkt nach der Anzahl der Leecher und fällt dann ebenfalls exponentiell ab. [8]

E. Von welchem Peer laden?

Der Tracker gibt bei einer Anfrage eine Liste von Peers zurück, welche an der Verteilung der Datei(en) beteiligt sind. Der Client versucht von allen anderen Clients, die im Besitz von Chunks sind, welche ihn interesssieren, herunterzuladen. Die Clients, welche meist in ganz normalen Haushalten lokalisiert sind, können eine geringe Uploadrate besitzen. Bei populären Dateien, welche sehr viele interessierte Clients haben, kann die Bandbreite der Clients unter Umständen nicht ausreichen, um alle anfragenden Clients zu bedienen. Der Choking Algorithmus entscheidet für jeden anfragenden Client, ob seine Anfrage beantwortet (cooperate) oder ob seine Anfrage zurückgewiesen (choke) wird. Eine genauere Beschreibung wird in Kapitel V gegeben.

F. Unterbrechungsfreiheit

Um die Ausreizung der Übertragungsgeschwindigkeit zu erreichen ist es wichtig mehrere Anfragen parallel zu führen, weil eine Antwort des Clients unter Umständen etwas länger dauern kann. BitTorrent erreicht dies indem Chunks nochmal in 16 KByte große Teile, genannt Sub-Pieces, aufgeteilt werden. Der Client fordert nicht einen kompletten Chunk, sondern immer eine gewisse Anzahl an Sub-Pieces, meist fünf, parallel an. Sobald ein Sub-Piece fertig geladen wurde, wird sofort das Nächste angefordert. Dieses Verfahren erhöht die Kommunikation zwischen den Clients und hilft Fehler in der Übertragung zu vermeiden.

IV. WELCHER CHUNK?

Jeder Client ist selbst verantwortlich in welcher Reihenfolge er Chunks von welchem Client herunterlädt. Damit die Bandbreite so gut wie möglich ausgenutzt wird, hat BitTorrent eine Strategie, um die Reihenfolge der Chunks zu bestimmen. Diese Strategie besteht aus vier Teilen, welche zu unterschiedlichen Zeitpunkten eingesetzt werden. Der BitTorrent Client entscheidet dabei aber nicht nur zu seinem eigenen Vorteil, sondern handelt auch im Interesse des Netzes, also auch der Peers, welche von ihm downloaden.

A. Strict Priority

Sobald ein Sub-Piece eines Chunks angefordert wurde, werden auch alle übrigen Sub-Pieces dieses Chunks angefordert, bevor andere Sub-Pieces anderer Chunks angefordert werden. Dadurch werden unvollständige Chunks so schnell wie möglich komplettiert und die Wahrscheinlichkeit, dass die Quelle eines Chunks verschwindet, wird minimiert. Eine Quelle eines Chunks kann zum Beispiel verschwinden, wenn der Peer, der im Besitz dieses Chunks ist das Internet verlässt, oder, wenn ein Benutzer das Client-Programm beendet.

B. Rarest First

Zum Zeitpunkt an dem ein Chunk vollständig geladen wurde, steht der Client vor der Wahl, welchen Chunk er als nächstes anfordert. Rarest First besagt hier, dass der Client den Chunk wählt, der in der geringsten Anzahl auf den anderen Peers vorhanden ist. Diese Strategie sorgt dafür, dass der Client immer Chunks hat, welche von vielen seiner Peers im selben Netz angefordert werden. Somit kann er immer uploaden, wodurch seine Upload-Bandbreite gut ausgenutzt wird.
Ein Torrent Netz kann vor folgenden Problemen stehen:

1) zu Beginn des Downloads: Wenn zum Beispiel eine Datei über ein Torrent Netzwerk verfügbar gemacht werden soll und dafür zu Beginn nur ein Seed eingesetzt wird, hat man das Problem, dass alle interessierten Clients von diesem Seed laden möchten. Man hat also eine herkömmliche Client - Server Situation. Rarest First garantiert hier, dass jeder Interessierte einen anderen Chunk lädt und sich die Quellen für jeden Chunk somit rasch vervielfachen, was die Downloadgeschwindigkeit für zukünftig anfordernde Peers steigert.

2) Der Start-Seeder wird heruntergefahren: Der Seed, mit welchem der Upload gestartet wurde, kann, sei es aus Kostengründen, oder Wartungsgründen, vom Netz genommen werden. Nun ist es möglich, dass ein bestimmter Chunk nicht mehr verfügbar ist, weil alle anderen Peers, welche diesen Chunk ebenfalls besitzen, zufälligerweise offline sind. Der Torrent ist jetzt unbrauchbar, da ein vollständiger Download der Datei nicht mehr möglich ist. Diesem Problem, beugt Rarest First vor, da die unverbreitetsten Chunks so schnell als möglich vervielfältigt werden und so die Wahrscheinlichkeit einer solchen Situation zu begegnen, verringert wird.

C. Random First Piece

Rarest First liefert zum Zeitpuntk, an dem ein Peer mit dem Download einer Datei beginnt schlechtere Resultate. Der Peer hat zu dieser Zeit keine Chunks, welche für andere Peers von Interesse wären und kann demnach nicht uploaden. Würde er nun als erstes den Chunk runterladen, der am seltensten vorkommt würde der Download, im Verhältnis zum Download eines öfter vorkommenden Chunks, langsamer von statten gehen. Deshalb ist es wichtig für Peers, nach dem Downloadstart so schnell wie möglich einen vollständigen Chunk zu bekommen. Würde die Strategie in diesem Fall immer den schnellsten Peer wählen, würde dieser überlastet werden, da jeder neue Peer sich sofort auf ihn stürzen würde. Random First Piece, der Peer wählt zu Beginn einen zufälligen Chunk aus, hat hier gute Resultate gezeigt. Sobald der Client einen vollständigen Chunk hat wird mit der Strategie Rarest First fortgefahren.

D. Endgame Mode

Wenn ein Chunk von einem Peer mit einer langsamen Anbindung angefordert wird, ist das inmitten eines Downloads kein Problem, aber es kann die Beendigung eines Downloads verzögern, was unerwünscht ist. Um dieses Problem gegen Ende eines Downloads zu unterbinden startet der Client den Endgame Modus. Alle fehlenden Sub-Pieces werden von allen Peers im Torrent Netz angefordert. Sobald eines dieser Sub-Pieces erhalten wurde wird die Anfrage dieses Sub-Pieces mittels einer Cancel-Mitteilung an alle Peers zurückgezogen, um nicht zu viel Bandbreite zu verschwenden. Das Ende einer Datei wird somit immer schnell geladen. Zeitlich gesehen ist der Endgame Modus von nur sehr kurzer Dauer, wodurch dieses Verhalten die Bandbreite nicht zu sehr belastet.

V. Der Choking Anlgorithmus

Um die Downloadgeschwindigkeit zu steigern wird jeder Peer versuchen von so vielen Peers wie nur möglich herunterzuladen. Man kann sich leicht vorstellen, dass diese Strategie das Netzwerk, sowie auch die einzelnen Peers überlasten würde. Aus diesem Grund wurde der Choking-Algorithmus eingeführt. Angenommen Peer A möchte von Peer B herunterladen, dann kann B die Anfrage von A bedienen oder ignorieren. Ein Peer erlaubt immer nur einer gewissen Anzahl an anderen Peers, typischerweise vier, das Downloaden. Alle anderen werden abgelehnt. Dieses Annehmen und Ablehnen wird durch den Choking-Algorithmus geregelt. Er ist im eigentlichen Sinne nicht Teil des BitTorrent Protokolls, aber unbedingt nötig um die Effizienz des gesamten Systems zu gewährleisten. Der Algorithmus versucht durch eine Strategie namens Tit for Tat Pareto Effizienz herzustellen [8].

A. Pareto Effizienz

Pareto Effizienz, benannt nach dem Ökonom und Soziologen Vilfredo Pareto (1848-1923), spielt in Wirtschaftswissenschaften eine Rolle. Ein System, in dem zwei Parteien miteinander in Aktion treten ist pareto effizient, wenn es den zwei Parteien gelingt einen Austausch durchzuführen, der Beiden nützt. Der Austausch wäre hierbei das Up- und Downloaden von Chunks. In der Informatik ist die Suche nach Pareto Effizienz ein lokaler Optimierungs-Algorithmus, welcher nach einem Weg sucht, durch den zwei Parteien eine Vermehrung des gemeinsamen Gutes erreichen können. Diese Art von Algorithmen neigt dazu, zu einem globalen Optimum zu führen. Das bedeutet, dass die Handlungen eines Peers nicht nur ihm allein, sondern dem gesamten System zu einer Verbesserung verhelfen [9].

B. Entwicklung von Tit for Tat

Die Entwicklung von Tit for Tat ist mit dem Gefangenendilemma verknüpft, weshalb vorher kurz auf dieses eingegangen

wird. Das Gefangenendilemma ist ein Paradoxon, welches von zwei Individuen handelt, welche angeblich ein gemeinsames Verbrechen begangen haben. Sie werden getrennt voneinander eingesperrt und befragt. Beiden wird angeboten, durch Verrat des Anderen die eigene Haftstrafe zu mindern. Verrät Häftling A Häftling B, so bekommt Häftling A 1 Jahr und Häftling B 5 Jahre aufgebrummt und umgekehrt. Schweigen beide werden beide zu 2 Jahren verurteilt und verraten sie sich gegenseitig werden beide für 4 Jahre weggesperrt [10].

	A schweigt	A verrät
B schweigt	A: 2 Jahre, B: 2 Jahre	A: 1 Jahr, B: 5 Jahre
B verrät	A: 5 Jahre, B: 1 Jahr	a: 4 Jahre, B: 4 Jahre

Tabelle II
DIESE TABELLE ZEIGT DIE STRAFEN ZU DEN VERSCHIEDENEN ENTSCHEIDUNGSMÖGLICHKEITEN DER GEFANGENEN A UND B. [10]

Ist es nun besser seinen Mittäter zu verraten, oder zu schweigen?
Dieser Frage wollte Robert Axelrod auf den Grund gehen. Dazu schrieb er einen Wettbewerb um die beste Strategie für ein auf 200 Runden basierendes Gefangenendilemma aus. Eine Runde besteht dabei aus jeweils einer Aktion von Gefangenem A und Gefangenem B. Die Jahre Gefängnis, welche der jeweilige Verbrecher in einer Runde bekommt werden aufsummiert. Ziel ist es natürlich am Ende so wenig wie möglich Jahre zu bekommen. Die beste Strategie in diesem Wettbewerb war überraschenderweise auch die einfachste. Diese Strategie wurde von Anatol Rapoport eingebracht und hat den Namen ”Tit for Tat” [11].

C. Tit for Tat

Der Begriff „Tit for Tat (TFT)“, oder auf deutsch „wie du mir, so ich dir“, kommt von der Spieltheorie und bezeichnet eine Strategie in der ein Spieler mit seinem Zug auf einen vorangegangenen Zug seines Gegenspielers antwortet.
Unter Betrachtung eines beliebigen, zugbasierten Spiels in dem Spieler A gegen Spieler B antritt, kann man sich TFT folgendermaßen erklären (angenommen Spieler B wendet TFT an): Spieler A handelt entgegen den Zielen von Spieler B, so wird Spieler B in seinem nächsten Zug ebenfalls entgegen den Interessen von Spieler A handeln. Profitiert Spieler B hingegen aus dem Zug von Spieler A, so wird auch Spieler B in seinem nächsten Zug entgegenkommend handeln.
Bei TFT handelt es sich um eine freundliche Strategie, was bedeutet, dass der erste Zug immer kooperierend ist. Angenommen zwei TFT Spieler begegnen sich, so kooperieren diese während des gesamten Spiels.

Abbildung 4. Das Händeschütteln am Beginn einer Konversation kann ein initiales Kooperieren darstellen [12]

TFT hat nach Axelrod vier wichtige Elemente, welche jede wirkungsvolle Strategie im wiederholten Gefangenendilemma auszeichnen. [11]

Klarheit	TFT ist so klar und einfach wie nur möglich.
Nettigkeit	TFT kooperiert im ersten Zug und auch in folgenden, so lange kein Verrat vorliegt.
Provozierbarkeit	TFT bestraft jeden Verrat sofort.
Nachsichtigkeit	TFT ist bereit die Kooperation sofort wieder aufzunehmen.

Tabelle III
VIER EIGENSCHAFTEN WELCHE JEDE WIRKUNGSVOLLE STRATEGIE IM WIEDERHOLTEN GEFANGENENDILEMMA AUSZEICHNEN [11].

D. Funktionsweise vom Choking-Algorithmus

Eine Kooperation im Sinne von Tit for Tat stellt im BitTorrent System einen Upload eines Chunks zu einem anderen Peer dar. Nicht kooperieren würde eine Ablehnung dieses Uploads bedeuten. Diese Kooperation und nicht Kooperation wird anhand der Downloadrate bestimmt. Peer A lädt von einer gewissen Anzhal von Peers jeweils mit einer gewissen Geschwindigkeit. Eine bestimmte Zahl anderer Peers wiederum möchten von Peer A herunterladen. Peer A muss nun aus diesem Pool vier auswählen, mit welchen er kooperiert. Peer A kontrolliert nun zuerst, ob auch er von diesen Peers herunterlädt und berechnet die Downloadgeschwindigkeit. A wird mit diesen Peers kooperieren, von welchen er am schnellsten herunterlädt. Lädt er von keinem herunter, passiert die Auswahl zufällig. Die Bestimmung der Downloadrate gestaltet sich als nicht ganz so einfach, wie man zunächst annehmen würde, weil die

Übertragungsrate relativ starken Schwankungen unterliegen kann. Die Downloadrate eines Peers könnte zum Zeitpunkt, in dem er angenommen wurde kurzzeitig gut sein und mit der Zeit einbrechen. Um zu verhindern, dass die Verbindung mit diesem Peer zu lange aufrechterhalten bleibt, berechnet ein Peer seine angenommenen Peers alle 10 Sekunden neu. 10 Sekunden reichen aus um neuen TCP-Verbindungen die Chance zu geben ihre volle Übertragungsrate zur Geltung zu bringen. Die Downloadrate kann anhand bereits laufender Downloads bestimmt werden. An diesem Punkt kann man die Analogie zu Tit for Tat erkennen. Lädt Peer A von Peer B so darf auch Peer B von Peer A laden [8].

E. Snubbed?

Wenn sich nun A für die schnellsten vier entschieden hat, kann es vorkommen, dass einem fünften Peer B der Download verweigert wird, obwohl er zu A uploadet. Es kann vorkommen, dass an Peer B nur vier Anfragen ankommen, was bedeutet, dass diese alle ohne Beachtung der Kriterien angenommen werden. B wird A also nicht dafür bestrafen, dass A den Download verweigert. Um in dieser Situation dennoch eine Bestrafung einzuführen handelt das BitTorrent System folgendermaen. Ein Peer betrachtet sich als abgewiesen (snubbed) von einem anderen Peer, wenn nach über einer Minute immer noch keine Verbindung zustande kommt. In diesem Fall wird auch Peer B die Verbindung zu Peer A unterbrechen. Wenn nun dasselbe auf der Seite von A passiert, wäre die Verbindung zwischen diesen beiden Punkten des Netzes verloren, weil nach Tit for Tat keine Möglichkeit mehr besteht, wie die Verbindung wieder aufgenommen werden kann, was ein erheblicher Nachteil für das gesamte Netzwerk darstellen kann. [8].

F. Optimistic Unchoke

Aus oben beschriebenem Grund hat der Choking Algorithmus eine Erweiterung zu Tit for Tat. Jeder Peer wählt zusätzlich zu den vier Peers, welche er annimmt, einen zusätzlichen aus dem Pool anfragender Peers aus, welchen er annimmt, ohne die Downloadgeschwindigkeit zu ihm zu überprüfen. Es handelt sich hierbei um eine otimistische Annahme (optimistic unchoke). Diese optimistische Annahme wird alle 30 Sekunden getätigt und bleibt auch für diese 30 Sekunden erhalten. Ein weiterer Grund für die optimistische Annahme ist, dass der erste Teil des Choking-Algorithmus keine Möglichkeit bietet neue, schnellere Verbindungen zu finden. Auch dies soll hierdurch gelöst sein [8].

G. Upload Only

Sobald ein Peer eine Datei komplett heruntergeladen hat, und sein Client-Programm nicht durch den Benutzer beendet wird, bietet er die Chunks weiterhin zum Upload an. Da er nicht mehr runterlädt hat er zu diesem Zeitpunkt allerdings keine Downloadraten mehr als Eingaben für den Choking-Algorithmus. BitTorrent betrachtet dann nicht mehr die Download- sondern die Uploadraten. Dadurch wird die volle Uploadkapazität ausgenutzt [8].

VI. Zusammenfassung und Ausblick

In den ersten Jahren wurde BitTorrent von der Medienindustrie als Feind angesehen, Trackerseiten wurden ausgeschalten, ein regelrechter Krieg gegen P2P-Systeme wurde eröffnet. Dem ist heute nicht mehr so. Firmen Präsident Ashwin Navin erklärt, dass der bereits 2005 fertig entwickelte Delivery Network Accellerator (DNA) erst 2007 veröffentlicht wurde da das Unternehemen vorher in einem schlechten Licht stand. Erst als BitTorrent mit 50 Medien Unternehmen im BitTorrent Entertainment Network Frieden geschlossen hatte, sah sich das Unternehmen bereit DNA zu veröffentlichen. DNA ermöglicht es Downloads in Teilen über ein P2P System abzuwickeln und verteilt dadurch die Last auf die Nutzer. Es wird zum Beispiel bei Video-Streaming eingesetzt.
Ab Version 6.0 ist BitTorrent Closed Source und daran soll sich auch nichts mehr ändern. Dieser Schritt sei laut BitTorrent-Präsident Ashwin Navin nötig geworden, da es öfters zu Problemen mit Drittanbietern gekommen sei. Diese sollen BitTorrent teilweise um Spyware ergänzt und unter eigenem Namen vertrieben haben.
Als Schwäche von BitTorrent könnte man anführen, dass nur wenig gegen das Verschwinden von unpopulären und älteren Dateien unternommen wird. Dies ist laut BitTorrent-Entwickler Bram Cohen aber auch Teil der Philosophie hinter BitTorrent: die populärsten Inhalte sollen sich am schnellsten verbreiten. In Zukunft soll aber wohl der Hauptclient (Client, der der erste Seeder einer Datei ist) Inhalte länger behalten, bzw verstärkt Inhalte anbieten, die zu verschwinden drohen.
Mit über 160 Millionen installierten Clients bietet BitTorrent ein mächtiges Werkzeug um Inhalte zu verbreiten. Seine Einfachheit hat sicherlich stark an der Verbreitung mitgewirkt. Aber nicht nur die Einfachheit sondern auch Eigenschaften wie Lastverteilung und Performance lassen BitTorrent zu einem der effektivsten P2P Systeme werden.
[13], [14]

VII. Begriffserklärung

Tracker: Server-Programm, welches es den Clients ermöglicht sich untereinander zu finden.
Chunk: BitTorrent teilt Dateien in kleinere Einheiten, genannt Chunks ein.
Sworm: Menge der Clients, welche an der selben Datei interessiert sind.
Seeder: Client, welcher eine vollständige Datei besitzt.
Peer: Client, der eine Datei sowohl verteilt, als auch herunterlädt.
Leecher: Zu diesem Begriff gibt es verschiedene Definitionen. Manche Quellen geben an es handle sich um einen Client, der nur herunterlädt und nicht weiterverteilt [6], andere geben an es sei dasselbe wie ein Peer [7].
.torrent: Datei, welche Informationen über die Datei, sowie über den Tracker enthält
choke, Choking: Die Zurückweisung eines anfragenden Clients
Sub-Piece: Teil eines Chunks
Pareto-Effizienz: Zustand, in dem es nicht möglich ist, ein

Individuum besser zu stellen, ohne zugleich ein anderes Individuum schlechter zu stellen.
snubbed: Zustand, den ein Peer einem anderen zuweist, wenn er ihm länger als eine Minute den Download verweigert.

LITERATUR

[1] BitTorrent.org, "Bittorrent.org," http://www.bittorrent.org, 2009.
[2] B. Cohen, "The bittorrent protocol specification," http://bittorrent.org/beps/bep_0003.html, 2008.
[3] J. Ihlenfeld, "Bittorrent überholt edonkey," http://www.golem.de/0610/48522.html, 2006.
[4] BitTorrent.Inc., "Bittorrent inc." http://www.bittorrent.com/, 2009.
[5] Wikipedia.org, "List of bittorrent clients," http://en.wikipedia.org/wiki/List_of_BitTorrent_clients, 2009.
[6] ——, "Leechen," http://de.wikipedia.org/wiki/Leecher, 2009.
[7] ——, "Leech," http://en.wikipedia.org/wiki/Leech_(computing), 2009.
[8] B. Cohen, "Incentives build robustness in bittorrent," http://bittorrent.org/bittorrentecon.pdf, May 2003.
[9] Wikipedia.org, "Pareto efficiency," http://en.wikipedia.org/wiki/Pareto_efficiency, 2009.
[10] A. Arbia, "Gefangenendilemma," http://www.scienceblogs.de/zoonpolitikon/2008/04/spieltheorie-einfach-erklart-i-einleitung-und-gefangenendilemma.php, 2008.
[11] C. Meredith, "Tit for tat," http://www2.owen.vanderbilt.edu/mike.shor/Courses/GTheory/docs/Axelrod.html, 1998.
[12] Wikipedia.org, "Tit for tat," http://en.wikipedia.org/wiki/Tit_for_tat, 2009.
[13] J. Ihlenfeld, "Bittorrent setzt künftig auf closed source," http://www.golem.de/0708/54016.html, 2007.
[14] ——, "Bittorrent richtet sich neu aus," http://www.golem.de/0710/55486.html, 2007.