Erkennung und Vermeidung von unkooperativem Verhalten in Peer-to-Peer-Datenstrukturen

Erik Buchmann

Universität Karlsruhe (TH)
Institut für Programmstrukturen und Datenorganisation
buchmann@ipd.uni-karlsruhe.de

Abstract: Peer-to-Peer (P2P)-Datenstrukturen (auch bekannt als P2P-Overlays oder Strukturierte P2P-Netzwerke) sind in der Lage, riesige Bestände an (Schlüssel,Wert)-Paaren effizient zu verwalten und dabei viele parallele Abfragen zu unterstützen. Dies wird erreicht, indem Datenbestand und Anfragelast auf alle Teilnehmer des Systems verteilt werden.

Vorschläge für P2P-Datenstrukturen gehen davon aus, dass die Peers ihren Teil der Anfragelast stets protokollgerecht verarbeiten. Für rationale Peers besteht die ökonomisch dominante Verhaltensweise jedoch darin, Anfragen anderer Teilnehmer nicht zu verarbeiten, sich also unkooperativ zu verhalten. Existierende Vorschläge zum Umgang mit unkooperativen Teilnehmern skalieren zumeist schlechter als die P2P-Datenstruktur selbst, lassen sich angreifen oder umgehen, oder basieren auf Annahmen, die einen Einsatz in der Praxis nicht zulassen.

Im Rahmen dieses Beitrags wird *FairNet* vorgestellt, ein Protokoll, das unkooperatives Verhalten ökonomisch unattraktiv macht. FairNet beruht darauf, dass topologisch benachbarte Peers nachvollziehbare Beobachtungen über geleistete oder verweigerte Arbeit austauschen. Peers, über die zu wenige positive Beobachtungen vorliegen, müssen einen Arbeitsbeweis erbringen, bevor sie am P2P-Netz partizipieren können. Das Protokoll führt dabei zur Entstehung von logischen transitiven Netzen von kooperativen Peers, die unkooperative Knoten von der Anfrageverarbeitung ausschließen und den Arbeitsbeweis als Eintrittsbarriere verwenden.

1 Einleitung

Peer-to-Peer (P2P)-Datenstrukturen sind selbstorganisierende, wartungsfreie Strukturen, die sehr große Bestände an (Schlüssel,Wert)-Paaren so verwalten können, dass sie von vielen Teilnehmern (den Peers) effizient parallel abgerufen werden können. Dies wird dadurch erreicht, dass die Daten mit dezentralen Algorithmen über alle Peers im System verteilt werden. Jeder Peer, der Anfragen an den Datenbestand stellt, bringt auch eigene Ressourcen (Speicher, CPU-Zeit, Netzwerkbandbreite) zur Anfrageverarbeitung in das System ein. Ein Teilnehmer der P2P-Datenstruktur startet dazu ein P2P-Programm, das eigene Anfragen an die Peers weiterleitet, die sie beantworten können, und gleichzeitig als Hintergrundprozeß die Anfragen von anderen Teilnehmern bearbeitet. Aus mehreren Gründen haben Peer-to-Peer-Datenstrukturen das Potential, zu einer Schlüsseltechnologien für zahlreiche im Internet angebotene Dienste zu werden:

- Die Zahl der Teilnehmer im Internet ist enorm. Google wird beispielsweise von 380 Mio. Benutzern pro Monat[1] verwendet. Derart populäre Web-Dienstleistungen erfordern große und teure Rechenzentren mit ausreichend Reservekapazität für zukünftige Anwendungen. Damit ist die Markteintrittsbarriere für neue Firmen, die den Massenmarkt adressieren, sehr hoch. Ein skalierendes System, bei dem jeder neue Benutzer so viele eigene Ressourcen beisteuert wie er verbraucht, erlaubt auch ambitionierten Risikoprojekten den Markteintritt.

- Das Internet weist derzeit einige problematische marktwirtschaftliche Anomalien auf. Beispielsweise resultiert das Spam-Problem[2] daraus, dass das Versenden von E-Mails nahezu kostenlos ist. Der Empfänger muss jedoch teure Server vorhalten. Eine andere Anomalie besteht darin, dass im WWW der Anbieter von Informationen für deren Bereitstellung bezahlen muss (Infrastrukturkosten, Server, Netzwerk-Kapazität); für den Kunden ist der Abruf von Webseiten hingegen kostenlos. Daher versuchen viele Anbieter, die Infrastrukturkosten durch den (oft unmäßigen) Einsatz von Werbung zu decken. Dienste, die auf der Basis von P2P-Datenstrukturen implementiert sind, verteilen die Arbeitslast auf ihre Nutzer und können derartige Marktanomalien vermeiden.

- Marktübliche PCs sind so leistungsfähig, dass sie einen Großteil ihrer Zeit untätig im Wartezustand verbringen. Die Verarbeitung von Anfragen in P2P-Datenstrukturen verursacht keinen messbaren Verschleiß, und Ressourcen wie z.B. Netzwerkbandbreite oder CPU-Zeit lassen sich auch nicht ansparen oder aufbrauchen. Mit P2P-Datenstrukturen gelingt es, diese brachliegenden Ressourcen nutzbar machen, beispielsweise indem ein P2P-Programm als Hintergrundprozeß abläuft.

P2P-Datenstrukturen [RFH+01, Abe01, SMLN+01] haben bereits einen hohen Entwicklungsstand erreicht, was technische Kriterien wie Anfrageverarbeitung, Selbstorganisation oder Ausfallsicherheit anbelangt. Diese Ansätze gehen jedoch davon aus, dass jeder Peer *freiwillig* eigene Ressourcen zur Verfügung stellt bzw. einen Teil der Anfragelast übernimmt. Untersuchungen in etablierten P2P-Systemen [AH00] haben jedoch gezeigt, dass diese Annahme nicht realistisch ist: die Teilnehmer versuchen, ihre Infrastrukturkosten zu minimieren. Wenn die Peers aber nicht bereit sind, *kooperativ* an der Anfrageverarbeitung mitzuwirken, sinkt sowohl die Verfügbarkeit des Systems als auch die Motivation der anderen Teilnehmer, ihre eigenen Ressourcen einzubringen.

Dieser Beitrag stellt ein Protokoll namens *FairNet* zum Umgang mit unkooperativen Teilnehmern in Peer-to-Peer-Datenstrukturen vor. FairNet hat die Aufgabe, unkooperatives Verhalten in einer P2P-Datenstruktur unattraktiv zu machen. Peers, die mittels unkooperativem Verhalten versuchen, ihre Infrastrukturkosten zu senken, sollen dabei derart sanktioniert werden, dass die durch Strafmaßnahmen verursachten Kosten die Einsparungen bei weitem übersteigen. Der Umgang mit unkooperativen Teilnehmern in P2P-Datenstrukturen ist aus mehreren Gründen schwierig. So müssen die technischen Eigenschaften der P2P-Systeme erhalten bleiben. Ein Protokoll, das unkooperative Peers diszipliniert, darf daher

[1] http://www.google.com/corporate/facts.html
[2] Unerwünschte E-Mail-Reklame

nicht auf einer zentralen Komponente (Single Point Of Failure, zentrales Angriffsziel, etc.) aufbauen, sondern muss mit den lokalen Beobachtungen und Interaktionen der Peers untereinander auskommen. Des weiteren passen sich unkooperative Peers an jede Gegenmaßnahme an. Das bedeutet, dass unkooperative Peers ihre Absichten verschleiern. Sie können versuchen, einen geringen Teil der Anfragen kooperativ zu beantworten, um nicht erkannt zu werden. Entdeckte unkooperative Teilnehmer können durch Aus- und Wiedereinsteigen in das P2P-System ihre Identität wechseln und ihr Verhalten mit einem "unverbrauchten" Pseudonym an anderer Stelle fortsetzen. Zuletzt muss es für kooperative Peers rational sein, das Protokoll auch einzusetzen. So müssen die Kosten des Protokolls für kooperative Knoten geringer sein als die Mehrkosten, die ihnen durch unkooperative Peers entstehen. Da Menschen im Allgemeinen risikoavers sind [KZ04], muss zudem die Wahrscheinlichkeit vernachlässigbar gering sein, dass ein kooperativer Teilnehmer fälschlich mit Strafen belegt wird.

FairNet ist in allen P2P-Datenstrukturen einsetzbar, in denen die Peers zahlreiche kleine Anfragen über einen längeren Zeitraum untereinander austauschen und beantworten. Es ist jedoch nicht verwendbar, wenn Garantien über die Verfügbarkeit von Daten oder Diensten benötigt werden, wenn der Verlust einzelner Anfragen nicht akzeptabel ist, oder wenn Peers nur für einen kurzen Zeitraum im System verbleiben und zahlreiche angesammelte Anfragen auf einmal stellen können. Des weiteren ist FairNet nicht als Maßnahme gegen unerwünschtes Verhalten geeignet, das nicht der Einsparung von Infrastrukturkosten dient. Gegen Denial-of-Service-Attacken oder andere Angriffe müssen also andere Lösungen gefunden werden.

2 P2P-Datenstrukturen

P2P-Datenstrukturen (z.B. Content-Addressable Networks [RFH+01], P-Grid [Abe01], Chord [SMLN+01] oder Pastry [RD01]) sind zur effizienten Speicherung von *(Schlüssel, Wert)*-Paaren entwickelt wurden. Alle Peers sind gleichberechtigt, d.h. jeder Peer darf Anfragen nach allen Daten in der Struktur absenden. Diese Daten werden über eine Programmierschnittstelle verwaltet, die einer Hashtabelle entspricht. Es werden Operationen wie `put(Schlüssel,Wert)` oder `Wert = get(Schlüssel)` angeboten. Daher werden P2P-Datenstrukturen auch häufig als *Verteilte Hashtabellen* (engl. *Distributed Hash Tables*, *DHT*) bezeichnet, selbst wenn sie intern auf einer Baumrepräsentation aufbauen [Abe01]. Ebenso wie in klassischen Hashtabellen unterstützen auch DHTs ausschließlich Operationen, die sich über einen Schlüssel adressieren lassen. Zur Verbreitung von Informationen nutzen P2P-Datenstrukturen physikalische Netzwerke wie z.B. das Internet, in denen eine Operation von Peer zu Peer weitergeleitet wird, bis sie einen Peer erreicht, der sie bearbeiten kann [AAG+05].

Zur Veranschaulichung und Evaluierung des FairNet-Protokolls wird in diesem Beitrag ein *Content-Addressable Network* (CAN) [RFH+01] verwendet. Ein CAN benutzt einen d-dimensionalen kartesischen Schlüsselraum $[0;1]^d$, der als Hyper-Torus organisiert ist. Demzufolge ist ein Schlüssel k eine kartesische Koordinate in diesem Schlüsselraum $k = (k_1, k_2, \cdots, k_d)$, $k \in [0;1]^d$. Jeder Peer ist für eine bestimmte Zone im Schlüsselraum

verantwortlich. Weiterhin kennt er alle Knoten, deren Zone im Schlüsselraum an seine angrenzt, d.h., er speichert Informationen über die Zonen sowie über die Netzwerkadressen seiner Nachbarn in seiner *Kontaktliste*. Neue Peers werden in das CAN aufgenommen, indem ein zufällig bestimmter Peer eine Hälfte seiner Zone sowie die darin enthaltenen (Schlüssel,Wert)-Paare abgibt. Abbildung 1 zeigt ein Beispiel für ein zweidimensionales CAN mit einem auf diese Weise entstandenen Zonenlayout.

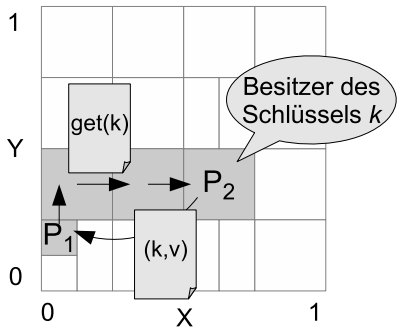

Abbildung 1: Nachrichtenweiterleitung im CAN.

Anfragen werden durch *Punkte* im Schlüsselraum adressiert. Erhält ein Peer eine Anfrage, dessen Schlüssel nicht in seine Zone gehört, so leitet er sie an einen geeigneten Nachbarn weiter. Dazu bestimmt der Peer den euklidischen Abstand zwischen dem Schlüssel der Anfrage und der Zone aller seiner Nachbarn. Ein Nachbar, dessen Zone den geringsten Abstand zum Schlüsselpunkt aufweist, erhält die Anfrage. Abbildung 1 zeigt schematisch, wie die Anfrageverarbeitung im CAN abläuft. Eine Anfrage nach einem Schlüssel k wird von Peer P_1 ausgehend so lange in Richtung des Zielpunkts k weitergeleitet, bis sie den Peer P_2 erreicht, der für die Zone verantwortlich ist, in die k passt. Der Peer P_2 beantwortet die Anfrage nun mit dem (Schlüssel,Wert)-Paar (k, v). Zu den Vorteilen des CAN zählt die Möglichkeit, Nachrichten um unzuverlässige Knoten herumzuleiten. Durch den mehrdimensionalen Schlüsselraum gibt es stets mehrere Möglichkeiten, einen bestimmten Datenbereich zu erreichen.

3 Unkooperatives Verhalten im CAN

Wenn auch nur einer der Weiterleiter eine Anfrage verwirft, anstelle sie zu beantworten oder geeignet weiterzugeben, bleibt die Anfrage unbeantwortet. Der Anfragesteller kann dann die Anfrage nur wiederholen und hoffen, dass sie über andere Peers weitergeleitet wird. Im Rahmen dieses Beitrags wird ein *unkooperativer Peer* [RL03] wie folgt definiert:

Ein unkooperativer Peer ist ein rationaler Teilnehmer einer P2P-Datenstruktur, der Ressourceneinsparungen zu realisieren versucht, indem er das Protokoll der Datenstruktur nicht befolgt.

Die Fokussierung auf wirtschaftlich rationales Verhalten grenzt dabei das unkooperative Verhalten von anderen Arten unerwünschten Verhaltens (z.B. Denial-of-Service-Angriffe oder das Einspeisen gefälschter Daten) in P2P-Datenstrukturen ab. Ein rationaler Peer wird sich sofort kooperativ verhalten, wenn er feststellt, dass er durch unkooperatives Betragen mit einem im Vergleich höheren Ressourcenbedarf beaufschlagt wird.

Unkooperative Peers führen dazu, dass die Verfügbarkeit der Daten und des Systems sinkt. Die Antwortzeit verlängert sich durch zu wiederholende Anfragen, und die Motivation der anderen Teilnehmer, selbst Ressourcen einzubringen, sinkt. Beispielsweise wird in einem CAN mit $N = 10.000$ Knoten und $d = 4$ Dimensionen eine Anfrage durchschnittlich $l = d/4 \cdot N^{1/d} = 10$ mal weitergeleitet. Angenommen, in diesem CAN befinden sich $u = 500$ unkooperative Teilnehmer, die eingehende Nachrichten verwerfen, ohne sie zu bearbeiten. Dann beträgt die Wahrscheinlichkeit, ein Ergebnis auf eine Anfrage zu erhalten, nur noch $(1 - u/N)^l \approx 60\%$.

4 Das FairNet-Protokoll

FairNet soll das kooperative Verhalten der Peers für alle Arten von Operationen (Anfrage, Einfügen, Löschen etc.) sicherstellen. Für eine vereinfachte Präsentation werden im Folgenden ausschließlich *Anfragen* betrachtet. FairNet kann aber für alle Arten von DHT-Operationen eingesetzt werden, bei denen der Absender der Operation das Ergebnis verifizieren kann. Der Anfrageverarbeitung in FairNet wird der folgende Ablauf zugrunde gelegt:

1. Der Anfragesteller generiert eine Anfrage, die mit einem Schlüssel adressiert ist.
2. Kann ein Peer eine Anfrage nicht aus seiner Zone beantworten, sendet er sie an einen *vertrauenswürdigen* Kontakt weiter, dessen Zone dem Anfrageschlüssel am nächsten ist.
3. Der Peer, in dessen Zone der Anfrageschlüssel passt, sendet die Antwort auf direktem Weg zum Anfragesteller zurück.
4. Erhält der Anfragesteller eine korrekte Antwort, so generiert er positives Feedback über den Peer, an den er die Anfrage zuvor weitergeleitet hatte, und sendet eine positive Feedback-Benachrichtigung an ihn. Erhält der Anfragesteller keine Antwort, erzeugt er negatives Feedback und sendet eine negative Feedback-Benachrichtigung. Wird die Kapazität des Repositories überschritten, wird das älteste Feedback-Objekt verdrängt.
5. Jeder Peer, der eine Feedback-Benachrichtigung erhält, erzeugt Feedback über den Peer, an den er vorher die Anfrage weitergeleitet hatte, und sendet die Feedback-Benachrichtigung an diesen weiter.

Die Peers leiten Anfragen im FairNet-Protokoll also nur an vertrauenswürdige Knoten weiter. Erhält ein Peer eine Information darüber, dass der Knoten, an den er selbst weitergeleitet hat, ebenfalls ordentlich gearbeitet hat, generiert er positives Feedback und gibt es an die Nachbarn dieses Peers weiter. Bei einer Information über eine nicht korrekte Bearbeitung einer Anfrage wird negatives Feedback generiert und weitergegeben.

Zwei Knoten kooperieren nur dann miteinander, wenn sie sich *gegenseitig* als kooperativ einschätzen. So wird ein Knoten P_1 eine Anfrage nur dann an einen Nachbarknoten P_2 weiterleiten, wenn P_1 Peer P_2 als kooperativ bewertet. Auf der anderen Seite wird P_2 diese Anfrage auch nur dann beantworten oder weiterleiten, wenn P_2 den Knoten P_1 für kooperativ hält. Ein Knoten P_1 wird einen anderen Knoten P_2 genau dann als kooperativ einschätzen, wenn das Repository von P_1 mindestens eine Anzahl von t *positiven* Feedback-Objekten mit Subjekt P_2 enthält.

Daraus erwächst jedoch ein Dilemma: wie können Neueinsteiger oder vormals unkooperative Knoten nachweisen, dass sie sich zukünftig kooperativ verhalten wollen? Zu diesem Zweck wird in FairNet ein *Arbeitsbeweis* eingeführt. Ein Arbeitsbeweis ist eine Aufgabe, die leicht zu formulieren ist, und dessen Lösung leicht überprüft werden kann. Die Lösung selbst ist jedoch nur schwer bzw. unter hohem Ressourceneinsatz zu finden [JJ99, Bac02]. Ein Beispiel ist dafür ist die Primzahlzerlegung großer Zahlen. Hat P_1 einen Arbeitsbeweis für P_2 erfolgreich erbracht, so schreibt P_2 eine Anzahl von positivem Feedback in sein lokales Repository. Für einen Peer ist nun es weniger ressourcenintensiv, sich über einen längeren Zeitraum kooperativ und protokollgerecht zu verhalten, als auch nur einen einzigen Arbeitsbeweis zu erbringen. Um die Zahl der Arbeitsbeweise zu verringern, welche die Peers erbringen müssen, tauschen benachbarte Knoten ihr Feedback untereinander aus Insgesamt bietet das hier vorgestellte Protokoll drei Alternativen dafür, dass ein Knoten P_2 positives Feedback über einen Nachbarn P_1 erhält bzw. selbst generiert:

R1 P_1 beantwortet eine Anfrage, die P_2 an ihn übermittelt hat, und P_2 erhält eine Benachrichtigung darüber, dass die Anfrage beantwortet wurde.

R2 P_1 erbringt einen Arbeitsbeweis, der von P_2 verlangt wurde. Hier kann P_2 unmittelbar beobachten, dass P_1 Arbeit geleistet hat.

R3 Die Nachbarn von P_1 gleichen ihr Feedback untereinander ab, und P_2 erfährt so davon, dass P_1 kooperative Arbeit für andere Peers geleistet hat.

Äquivalent dazu erhält P_2 negatives Feedback über P_1 in den folgenden Fällen:

R4 P_2 übermittelt eine Anfrage an P_1, die P_1 nicht beantwortet oder weiterleitet, und P_2 erhält eine Benachrichtigung darüber.

R5 P_1 leitet eine Anfrage von P_2 weiter, die einer der folgenden Weiterleiter nicht protokollgemäß verarbeitet, und P_2 erhält eine Benachrichtigung über eine unbeantwortete Anfrage.

R6 Nachbarn von P_1 tauschen ihr Feedback untereinander aus, und P_2 erhält auf diesem Wege negatives Feedback über P_1.

Allerdings sind die mit Feedback bedachten Operationen unterschiedlich 'teuer'. Beispielsweise benötigt das Weiterleiten verglichen mit einem Arbeitsbeweis viel weniger Ressourcen. Des weiteren ist es möglicherweise für eine effektive Disziplinierung der Peers wünschenswert, unkooperatives Verhalten mit mehr negativem Feedback zu bestrafen als kooperatives Verhalten zu belohnen.

Um diesen Sachverhalt abzubilden, wird nicht bei jedem Auftreten von **R1**, **R2**, **R4** oder **R5** ein Feedback-Objekt generiert. Stattdessen wird ein Feedback-Objekt mit einer Wahrscheinlichkeit erzeugt, die von der beobachteten Aktion abhängt. Die Wahrscheinlichkeiten repräsentieren also indirekt unterschiedliche Feedback-Gewichte. Wenn beispielswei-

se **R4** als doppelt so wichtig angesehen würde wie **R1**, bekäme **R4** auch eine doppelt so hohe Wahrscheinlichkeit für das Erzeugen von Feedback zugewiesen; folglich wäre auch doppelt so viel Feedback im Umlauf, welches **R4** zur Ursache hätte.

Eine ausführliche analytische Betrachtung, wie diese Gewichte bestimmt werden müssen, damit FairNet unkooperative Peers effektiv diszipliniert, würde den Rahmen dieses Beitrags sprengen, kann jedoch in [Buc06] nachgeschlagen werden. Im Folgenden sollen einige Ergebnisse dieser Betrachtungen präsentiert werden.

5 Evaluierung

Zur Evaluierung wird ein CAN mit $N = 10.000$ Teilnehmern und einem $d = 4$-dimensionalen Schlüsselraum verwendet. Jeder Peer hält ein Repository mit $s = 10$ Feedback-Objekten über jeden Nachbarn. Bei mindestens t positiven Feedback-Objekten gilt ein Peer als kooperativ. Variiert wird die globale Ausfallwahrscheinlichkeit p. Der Fall $p = 0$ steht für ein CAN, in dem jede Anfrage beantwortet wird. Eine globale Ausfallwahrscheinlichkeit von $p = 0{,}2$ bedeutet hingegen, dass jeder Peer im Durchschnitt jede 5. Nachricht verwirft, oder dass jeder 5. Peer keine einzige Nachricht korrekt verarbeitet.

Zuerst sollen **kooperative Knoten** betrachtet werden, die mit der *globalen Ausfallwahrscheinlichkeit* p Nachrichten nicht korrekt verarbeiten. Deren Kosten c hängen wesentlich vom Schwellenwert t und von der Ausfallwahrscheinlichkeit p ab. Abbildung 2 zeigt die Kosten für das Weiterleiten und Beantworten von Nachrichten, Abbildung 3 stellt die Kosten für das Erbringen von Arbeitsbeweisen dar.

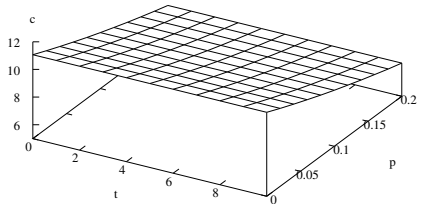

Abbildung 2: Kosten für reguläre Arbeit.

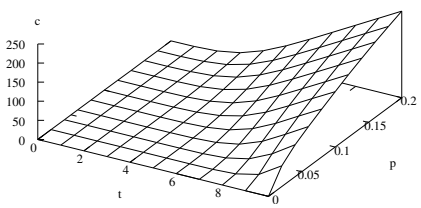

Abbildung 3: Kosten für Arbeitsbeweise.

Abbildung 2 zeigt, dass eine hohe globale Ausfallwahrscheinlichkeit die Kosten für das Beantworten und Weiterleiten für kooperative Knoten reduziert, da viele Nachrichten nicht beantwortet werden. Auf der anderen Seite erhöht sich dadurch die Zahl der zu erbringenden Arbeitsbeweise (Abbildung 3), insbesondere bei einem hohen Schwellwert t.

Die Kosten **unkooperativer Knoten** hängen ebenfalls vom Schwellenwert t, sowie von ihrer *lokalen Ausfallwahrscheinlichkeit* q ab. Um die Kosten unkooperativer Knoten mit denen von kooperativen Teilnehmern ins Verhältnis zu setzen, wird die lokale Ausfallwahrscheinlichkeit (willkürlich) auf auf $q = p \cdot 2$ festgelegt. Ein unkooperativer Knoten verarbeitet eine Nachricht also doppelt so häufig nicht korrekt, wie beliebige andere (kooperative) Knoten. Hierbei handelt es sich um eine sehr konservative Annahme, die nach-

weisen soll, dass selbst geringe Unterschiede in den Ausfallraten vom Protokoll erkannt und abgestraft werden. Bei einer kleinen globalen Ausfallwahrscheinlichkeit von $p = 5\%$ würde ein unkooperativer Peer immer noch 90% aller Nachrichten korrekt verarbeiten!

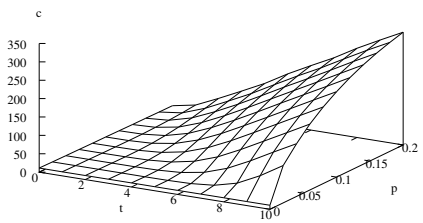

Abbildung 4: Gesamtkosten für unkooperative Peers.

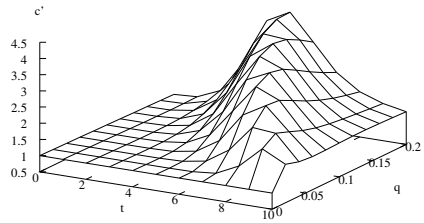

Abbildung 5: Diskriminierung unkooperativer Peers.

Abbildung 4 zeigt nun für unkooperative Peers die Summe der Kosten für Arbeitsbeweise, für das Weiterleiten und das Beantworten in Abhängigkeit vom Schwellenwert t und der globalen Ausfallwahrscheinlichkeit p. Beim Schwellenwert $t = 0$ wird jeder Knoten für kooperativ gehalten, und es werden keine Arbeitsbeweise verlangt. Abbildung 4 weist für diesen Fall erwartungsgemäß niedrige Gesamtkosten aus. Gemäß Abbildung 2 sinken die Gesamtkosten geringfügig mit steigender Ausfallwahrscheinlichkeit. Die dadurch erzielbaren Einsparungen sind jedoch zu vernachlässigen, wenn sinnvolle Schwellenwerte gewählt werden: mit steigendem Schwellenwert t nimmt die Zahl der Arbeitsbeweise rapide zu. Selbst bei so geringen Ausfallwahrscheinlichkeiten wie $p = 0{,}05$ und $q = 0{,}1$ werden ab $t \geq 5$ Arbeitsbeweise verlangt.

Um zu untersuchen, wann unkooperative Knoten höhere Gesamtkosten tragen müssen als kooperative Peers, wird die *Diskriminierung* c' definiert als der Quotient aus den Gesamtkosten der unkooperativen Peers dividiert durch die Gesamtkosten der kooperativen Peers: $c' = \frac{c_{unkoop}}{c_{koop}}$. Eine Diskriminierung von $c' = 2$ bedeutet, dass ein unkooperativer Peer in FairNet doppelt so hohe Gesamtkosten aufbringen muss wie ein kooperativer. Damit das Protokoll unkooperatives Verhalten unattraktiv macht, muss die Diskriminierung stets deutlich größer sein als 1. Abbildung 5 zeigt die Diskriminierung in Abhängigkeit vom Schwellenwert t und der Ausfallwahrscheinlichkeit q. Wieder gilt $q = p \cdot 2$. Bei kleinen Schwellenwerten t und geringen Ausfallwahrscheinlichkeiten werden sowohl von kooperativen als auch unkooperativen Knoten keine Arbeitsbeweise verlangt; die Kosten für beide Verhaltensweisen sind gleich und $c' = 1$. Bei mittleren Ausfallwahrscheinlichkeiten und Schwellenwerten um den Wert $t = 5$ kann eine Diskriminierung bis zu $4{,}5$ erreicht werden. Wird der Schwellenwert weiter erhöht, sinkt die Diskriminierung wieder, weil nun sowohl von kooperativen als auch von unkooperativen Knoten sehr viele Arbeitsbeweise verlangt werden. Die Abbildung weist nach, dass es selbst unter Extrembedingungen wie $t = 0$, $t = s$ oder einer hohen Ausfallwahrscheinlichkeit niemals vorkommt, dass die Diskriminierung kleiner 1 wird. Zum Anderen zeigt sie, dass es einen sehr großen Bereich gibt, in dem die Kosten für kooperative Peers moderat sind (vgl. Abb. 3), in dem die Diskriminierung aber so groß ist, dass unkooperative Knoten mindestens doppelt so viel leisten müssen wie kooperative.

6 Zusammenfassung

Unkooperative Peers, die Operationen nicht protokollgerecht verarbeiten um Ressourcen zu sparen, sind ein ernstes Problem für alle Arten von P2P-Datenstrukturen. Insbesondere das Fehlen von zentralisierten Strukturen, welche Aufgaben wie Monitoring, Authentifizierung oder Accounting übernehmen könnten, machen den Umgang mit unkooperativen Teilnehmern in P2P-Datenstrukturen zu einer Herausforderung. Hinzu kommt, dass die Peers jederzeit ihr Verhalten ändern oder das System verlassen und unter einer neuen Identität wieder eintreten können. In diesem Beitrag wurde *FairNet* vorgestellt, um unkooperatives Verhalten für rationale Peers unattraktiv zu machen. FairNet basiert darauf, dass Peers Beobachtungen über das Verhalten von anderen in Form von Feedback untereinander austauschen. Nur Peers, über die genug positives Feedback vorliegt, dürfen an der Anfrageverarbeitung teilnehmen. Alle anderen Peers müssen einen Arbeitsbeweis erbringen, bevor sie Anfragen absetzen oder Anfrageergebnisse erhalten können.

Dieser Beitrag erläutert das Kernkonzept der in [Buc06] präsentierten Arbeiten. Neben einer ausführlichen Darstellung der in diesem Beitrag vorgestellten Themen bietet [Buc06] darüber hinaus eine experimentelle Evaluierung, welche die Übertragbarkeit der analytischen Ergebnisse auf einen implementierten Prototypen und die Anwendbarkeit in der Praxis nachweist, eine experimentelle Untersuchung von möglichen Angriffen auf das Protokoll, eine effektive Erweiterung zum Umgang gefälschtem Feedback, Strategien zur aufwandsneutralen Feedback-Verbreitung als 'Anhängsel' an ohnehin versendete Datenpakete sowie eine Untersuchung der Anwendbarkeit von FairNet in anderen P2P-Datenstrukturen. Das Resultat dieser Arbeiten ist ein robustes, skalierbares und flexibel parametrisierbares Protokoll, das selbst unkooperative Teilnehmer, die noch 90% der eingehenden Anfragen korrekt beantworten, zuverlässig erkennt und mit Mehrkosten beaufschlagt, während kooperativen Knoten dabei durch das Protokoll nur 10% Mehrkosten entstehen.

Literatur

[AAG+05] Karl Aberer, Luc Onana Alima, Ali Ghodsi, Sarunas Girdzijauskas, Seif Haridi und Manfred Hauswirth. The Essence of P2P: A Reference Architecture for Overlay Networks. In *Proceedings of the 5th International Conference on Peer-to-Peer Computing (P2P'05)*, Seiten 11–20, 2005.

[Abe01] Karl Aberer. P-Grid: A Self-Organizing Access Structure for P2P Information Systems. In *Proceedings of the 9th International Conference on Cooperative Information Systems (CoopIS'01)*, Seiten 179–194, 2001.

[AH00] E. Adar und B. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), Oktober 2000.

[Bac02] Adam Back. Hashcash - A Denial of Service Counter-Measure. http://hashcash.org, August 2002.

[Buc06] Erik Buchmann. Erkennung und Vermeidung von unkooperativem Verhalten in Peer-to-Peer-Datenstrukturen. *Dissertationsschrift, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg*, 2006.

[JJ99] M. Jakobsson und A. Juels. Proofs of Work and Bread Pudding Protocols. In *Proceedings of the 4th International Conference on Communications and Multimedia Security (CMS'99)*, 1999.

[KZ04] Frederic Koessler und Anthony Ziegelmeyer. Parimutuel Betting under Asymmetric Information. Bericht 2003-34, Max Planck Institute of Economics, Strategic Interaction Group, Marz 2004. .

[RD01] Antony Rowstron und Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the 21st International Conference on Distributed Systems (ICDCS'01)*, 2001.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp und Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'01)*, August 2001.

[RL03] L. Ramaswamy und Ling Liu. Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems. *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, 2003.

[SMLN+01] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M. Frans Kaashoek, Frank Dabek und Hari Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'01)*, August 2001.

Erik Buchmann wurde am 30.07.1976 in Magdeburg geboren. Nach dem Studium der Fachrichtung "Wirtschaftsinformatik" an der Fakultät für Informatik der Otto-von-Guericke-Universität in Magdeburg promovierte Herr Buchmann im Jahr 2006 erfolgreich (Prädikat "summa cum laude", Dissertationspreis der Fakultät) an der selben Einrichtung. Er arbeitet heute als Oberassistent am Institut für Programmstrukturen und Datenorganisation der Universität Karlsruhe (TH). Seine wissenschaftlichen Interessen liegen in den Bereichen verteilter selbstorganisierender Kleinstrechner (Sensornetzwerke, RFID, Ubiquitous Computing) und deren Auswirkungen auf die Gesellschaft.