Anreizmechanismen für Peer-to-Peer Web Crawling unter Berücksichtigung bösartiger Teilnehmer

Holger Steinhaus

Institut für technische und betriebliche Informationssysteme
Otto-von-Guericke-Universität Magdeburg
39106 Magdeburg
hsteinha@cs.uni-magdeburg.de

Klemens Böhm, Stephan Schosser

Institut für Programmstrukturen und Datenorganisation
Universität Karlsruhe (TH)
76131 Karlsruhe
{boehm,schosser}@ipd.uni-karlsruhe.de

Abstract: Web Crawling ist grundlegend für eine Vielzahl von Anwendungen. Aufgrund der damit verbundenen hohen Ressourcenanforderungen sind jedoch nur wenige Organisationen in der Lage, eigene Crawler zu betreiben. Um neue Anwendungsbereiche zu erschließen und eine hohe Skalierbarkeit sowie eine faire Verteilung der Kosten zu erreichen, muss Web Crawling nicht nur verteilt, sondern auch koordinatorfrei sein. Peer-to-Peer-Techniken können diese Anforderungen erfüllen, benötigen jedoch Anreizmechanismen, um Teilnehmer nachhaltig zur Beteiligung zu motivieren.

Solche Mechanismen müssen jedoch einige Besonderheiten der Anwendung 'Web Crawling' berücksichtigen: Die Erkennung von sich nicht beteiligenden Teilnehmern ist schwierig, da diese nicht direkt beobachtet werden kann. Weiterhin sollte es einem Teilnehmer erlaubt sein, vorübergehend nicht zu crawlen, ohne in den Augen anderer Peers als bösartig betrachtet zu werden. Weiterhin müssen die zu entwerfenden Mechanismen robust gegen gezielte bösartige Angriffe sein. Wir stellen im Folgenden Anreizmechanismen für ein Peer-to-Peer-System vor, die diese Anforderungen erfüllen. Neben einem Feedback-basierten Reputationssystem kommen dabei Mechanismen zum Einsatz; die eine Überprüfung von Leistungsbehauptungen einzelner Teilnehmern ermöglichen. Umfangreiche Simulationsexperimente evaluieren die Auswirkungen bösartiger Teilnehmer und zeigen die Effektivität und Robustheit des Ansatzes.

1 Einleitung

1.1 Problembeschreibung

Eine Vielzahl von Webanwendungen, wie z.B. Preisvergleiche, Webarchive, Webservice-Lokalisierungsdienste oder spezialisierte Suchmaschinen bauen auf Webcawlern auf. Der Betrieb eines Webcrawlers ist jedoch äußerst aufwändig, wenn man einen signifikanten Teil des Webs vollständig und zeitnah abdecken will. Dazu kommen normalerweise große Rechnerfarmen zum Einsatz, die aus dem Crawling resultierenden Datenmengen stellen eine große Herausforderung für konventionelle verteilte Datenbanken dar[1]. Nur wenige große Unternehmen oder Organisationen sind in der Lage und bereit, solche Datenmengen zu bewältigen.

Unser Ziel ist es, Web Crawling einem breiteren Publikum zugänglich zu machen, u.a. kleineren Unternehmen, die neue crawling-basierte Dienste anbieten wollen, jedoch nicht die Ressourcen zum Crawlen des gesamten Webs aufbringen können. Die Grundidee besteht darin, dass viele verschiedene Teilnehmer und Organisationen gemeinsam crawlen, wenn ihre Ressourcen ansonsten brach liegen würden. Dazu ist eine gewisse Koordination erforderlich. Diese sollte jedoch nicht zentral erfolgen, da ein solches Modell im Allgemeinen schlecht skaliert und empfindlich gegenüber Denial-of-Service-Angriffen ist. Diese Koordinationsmechanismen müssen weiterhin einen Anreiz zur Beteiligung bieten, um *Free Riding* [AH00] zu verhindern. Dabei sollte jeder Teilnehmer die Kontrolle über seine Ressourcen behalten und selbst entscheiden können, wann er seine Bandbreite und Rechenleistung einsetzt. Folglich ist eine einfache Unterscheidung zwischen 'guten' und 'bösen' Teilnehmern bzw. zwischen gerade crawlenden und gerade nicht crawlenden zu kurz gegriffen. Weiterhin sollte berücksichtigt werden, dass sich in der Realität nicht alle Teilnehmer zwangsläufig rational verhalten. Die Erfahrung zeigt, dass Angreifer in offenen Umgebungen oft bereit sind, bedeutende eigene Ressourcen aufzubringen, um dem angegriffenen Dienst Schaden zuzufügen. Dieses Verhalten soll im Folgenden als *bösartig* bezeichnet werden. Die hier präsentierten Mechanismen haben daher zum Ziel, gegen beliebige Kombinationen aus Free Riding und bösartigkeit wirksam zu sein.

Im Folgenden stellen wir Anreiz- und Verteidigungsmechanismen für einen P2P-Webcrawler vor, die diese Fragestellungen abdecken. Wir stellen ein Protokoll vor, dass es einem Teilnehmer ermöglicht, die behauptete Crawlingleistung eines Anderen anhand eines stichprobenbasierten mehrstufigen Verfahrens zu überprüfen. Die Überprüfung ist jedoch nicht trivial, da verschiedene Teilnehmer unterschiedliche Bereiche des Webs crawlen. Anhand der Reputation des betreffenden Peers wird dabei die Frequenz solcher Überprüfungen gesteuert, was den gesamten durch Überprüfungen verursachten Overhead in Grenzen hält. Die Ergebnisse der Überprüfungen beeinflussen wiederum die Reputation des geprüften Teilnehmers. Das Überprüfungsprotokoll stellt den Kern der hier vorgestellten Arbeit dar, während die Realisierung der Reputationsverwaltung auf bewährte Ansätze

[1] Ein Vorabexperiment zur Analyse der Webstruktur wurde auf einer Breite von 12 Mio. gecrawlten Webseiten (ca. 0.1% der geschätzten Größe [GS05] des Webs im Jahr 2005) durchgeführt. Das Speichern der Linkstruktur der besuchten Seiten (ohne deren Inhalt) führte zu einer Datenbank mit 1 Mrd. Tupeln, die etwa 80 GB an permanentem Speicherplatz benötigt.

im Sinne von [AD01] zurückgreift. Wir evaluieren unseren Ansatz anhand umfangreicher Experimente mit verschiedenen Formen von Fehlverhalten. Diese zeigen, dass beispielsweise ein Denial-of-Service-Angreifer 15-30% aller Teilnehmer des Systems unter seine Kontrolle bringen muss, um einen signifikaten Schaden zu verursachen.

1.2 Related Work

P2P-Datenstrukturen haben in der Vergangenheit viel wissenschaftliches Interesse gefunden. [LNS93] stellte als eine der ersten Arbeiten eine vollständig dezentral organisierte verteilte Datenstruktur vor. Sogenannte *Overlay-Netzwerke* [R$^+$01, S$^+$01] sind Ansätze, die ähnlichen Zielen folgen[LNS93]. Allen Ansätzen ist jedoch gemein, dass sie unkooperatives oder bösartiges Verhalten von Teilnehmern nicht betrachten. Diese Gesichtspunkte sind Gegenstand der Betrachtungen in [BB06]. [AD01, LFSC03, FLSC04] betrachten unkooperatives Verhalten anhand des evolutionären Gefangenendilemmas. Die dort verwendete Definition von Unkooperativität ist jedoch deutlich enger gefasst, als für unser Szenario notwendig. [LFSC03, FLSC04] nehmen weiterhin Rationalität aller Teilnehmer als Voraussetzung an. Damit ist es jedoch nicht möglich, alle in der Realität vorstellbaren Bedrohungen abzubilden, insbesondere wenn Angreifer aufgrund einer externen Motivation zum Einsatz signifikanter eigener Ressourcen bereit sind.

Grundlegende Betrachtungen zum verteilten Webcrawling finden sich in [CGM02], dabei ist jedoch für die fortgeschritteneren der vorgestellten Ansätze eine zentralisierte Koordination notwendig. [HN99] beschreibt hingegen eine konkrete Implementierung eines verteilten Webcrawlers. Andere Ansätze wie UbiCrawler [BCSV04] sind vollständig dezentral organisiert. Sie gehen jedoch von einer vertrauenswürdigen geschlossenen Umgebung ohne unzuverlässige oder eigennützige Teilnehmer aus. Darüber hinaus existieren einige konkrete Implementierungen von P2P-artigen Webcrawlern wie z.B. Yacy und Odissea [Chr05, S$^+$03]. Nach unseren Erkenntnissen lassen diese ebenfalls die Aspekte Unkooperativität und bösartiges Verhalten außer Acht, deren Berücksichtigung uns jedoch unverzichtbar erscheint.

1.3 Annahmen

Die Effektivität unserer Lösung ist an einige Annahmen gebunden, die im Folgenden beschrieben werden:

Statische Zuweisung von Partitionen zu Teilnehmern. Jeder Teilnehmer ist für eine Partition des Webs verantwortlich. Eine Partition ist ein Teil des Schlüsselraums, der durch Anwendung einer Hashfunktion auf URLs aufgespannt wird. Die Partition wird also durch einen minimalen und einen maximalen Schlüsselwert eindeutig beschrieben. Die Funktion zur Berechnung und zum Vergleichen von Hashwerten ist allen Teilnehmern bekannt. Frühere Arbeiten zu strukturierten P2P-Datenstrukturen [R$^+$01, S$^+$01] beschreiben die technischen Aspekte der Zuordnung von Partitionen zu Teilnehmern.

Statisches Web. Das Web ist statisch, d.h. es werden keine Seiten während des Crawls hinzugefügt, gelöscht oder geändert. Diese Annahme steht dem Ziel eines High-Performance-Webcrawlers nicht im Wege, da dieser auf kurze Aktualisierungszyklen zielt, die deutlich unter der typischen Aktualisierungssrate im Web liegt.

Die Mehrheit der Teilnehmer ist kooperativ. Unsere Betrachtungen setzen einen Gleichgewichtszustand voraus, in dem eine Mehrheit der Peers den hier vorgestellten Verhaltensweisen folgt. Diese Teilnehmer werden als *kooperativ* bezeichnet. Diese Annahme ist nicht restriktiv, da erst eine solche Mehrheit die Definition eines 'Sollzustandes' des kooperativen Peers ermöglicht, andere Arbeiten wie z.B. [AD01] verfolgen ähnliche Ansätze.

Jeder Teilnehmer speichert seine durch Crawling gesammelten Daten selbst. Andere Peers können dann anschließend darauf zugreifen. Diese Annahme ist nicht Bestandteil unseres Ansatzes (der Speicherort der Daten ist durch die DHT völlig transparent), sondern dient lediglich der Fokussierung auf den Hauptgegenstand dieses Artikels. Die Anforderungen an die Datenspeicherung sind teilweise stark applikationsspezifisch. Ein P2P-Webarchiv wird z.B. den gleichen Schlüsselraum für Arbeitsverteilung (Partitionierung), Speicherung von Crawlingdaten und Anfragen verwenden (die URL der betreffenden Seite). Andere Applikationen, wie z.B. eine Suchmaschine, erfordern andere Datenorganisationen. Im Fall der Suchmaschine erfolgt die Partionierung wiederum anhand von Seiten-URLs, die Speicherung und Anfrage jedoch indiziert nach Stichwörtern.

Diskrete Nutzenfunktion. Der Nutzen eines Teilnehmers ist 1, wenn für den betreffenden Teilnehmer Anfragen an das System erlaubt sind, sonst 0. Der Nutzen hängt somit nicht von der Zahl der erlaubten Anfragen ab. Diese Annahme entspricht einem Flatrate-Abrechnungsmodell, das in vielen Bereichen erfolgreich eingesetzt wird.

2 Verteiltes Webcrawling

Aufgrund des Ressourcenbedarfs eines Webcrawlers und der Größe des Webs kann ein Crawler nur dann einen signifikanten Teil des Webs in endlicher Zeit abdecken, wenn die Arbeit auf eine große Zahl von parallel arbeitenden Knoten verteilt wird. [CGM02] bietet einen guten Überblick über die verschiedenen Ansätze zur Parallelisierung eines Webcrawlers, geht dabei jedoch größtenteils vom Vorhandensein eines zentralen Koordinators aus.

Unser Ansatz geht von einer partitionsbasierten Verteilung der Arbeitslast aus, die keine solche Zentralinstanz benötigt. Die Partitionierung wird mit Hilfe einer Hashfunktion erreicht, die URLs von Webseiten auf einen numerischen Bereich fester Länge abbildet[2]. Anhand dieses Wertes kann eine verteilte Hashtabelle (DHT) aufgebaut werden, beispielsweise unter Verwendung von Protokollen wie CAN [R$^+$01] oder Chord [S$^+$01]. Abgesehen von der allen Peers bekannten Hashfunktion ist keine darüber hinausgehende Koordination erforderlich. Jeder Teilnehmer kann lokal entscheiden, ob eine bestimmte URL zu seiner Partition gehört. URLs, die nicht zur lokalen Partition gehören, können mit Hilfe

[2]Wir verwenden dazu eine kryptographische Hashfunktion (MD5), die aufgrund ihres Entwurfszwecks auch sehr ähnliche Eingangswerte sehr gleichmäßig über den gesamten möglichen Wertebereich streut.

des DHT-Protokolls transparent auf den entsprechenden entfernten Teilnehmer abgebildet werden.

Die Partitionierung stellt weiterhin sicher, das einzelne Seiten nicht mehrmals gecrawlt werden, ohne dabei aufwändige verteilte Sperrmechanismen anwenden zu müssen. Die Vermeidung redundanten Crawlens ist wichtig, da nicht nur Ressourcen des Crawlers gebunden werden, sondern auch die des gecrawlten Webservers. Folglich betrachten viele Serverbetreiber redundantes Crawling als 'unfreundlich' und sperren solche Crawler mittels technischer Maßnahmen künftig aus.

3 Verhalten von Teilnehmern

3.1 bösartige vs. passive Teilnehmer

Im Kontext eines P2P-Systems wird das Verhalten von Teilnehmern insbesondere durch zwei Eigenschaften charakterisiert: Zunächst ist es von vitalem Interesse, alle Teilnehmer zu einer hinreichenden und nachhaltigen *Beteiligung* zu motivieren. Anfragen sind aus diesem Grund nur solchen Teilnehmern gestattet, die sich hinreichend beteiligt haben. Die Beteiligung ergibt sich dabei aus der Zahl der bislang gecrawlten Seiten. Die zweite wichtige Eigenschaft, die sich orthogonal zur Beteiligung verhält, ist die Kooperativität. Ein Teilnehmer wird kooperativ genannt, wenn sein Verhalten das System nicht schädigt. Das ist gegeben, wenn er die folgenden Regeln befolgt:

- Er übermittelt regelmäßig und wahrheitsgemäß Feedback über Erfahrungen mit anderen Teilnehmern
- Er beteiligt sich in der Pflege eines Reputation Repository.
- Er beteiligt sich an der Erkennung und Sanktionierung nicht-kooperativer Teilnehmer.
- Er beantwortet Anfragen von dazu berechtigten kooperativen Teilnehmern.

Passives Verhalten. Ein kooperativer Teilnehmer muss nicht ständig crawlen - das würde der Idee der Nutzung von Leerlaufzeiten widersprechen. Eine andere potentielle Ursache, warum ein Peer nicht crawlt, kann jedoch darin bestehen, dass der Eigentümer des Knotens sich im Free Riding versucht. Ohne geeignete Gegenmaßnahmen ist dieses Verhalten offensichtlich eine dominante Strategie, alle rationalen Teilnehmer würden früher oder später zu dieser Strategie übergehen - mit fatalen Folgen für das Gesamtsystem. Ein wichtiges Entwurfsziel war es daher, dass ein Free Rider keinen Nutzen erzielen kann: Nur eine hinreichende bisherige Beteiligung $> s$ ermöglicht einem Teilnehmer, selbst Anfragen zu stellen. Der exogene Parameter s wird in Abschnitt 3.3 näher betrachtet. Unter dieser Bedingung ist es für einen Teilnehmer rational, sich zu beteiligen. Alle Teilnehmer, deren Beteiligung s nicht erreicht, werden im Folgenden als *passiv* bezeichnet.

bösartiges Verhalten. Nicht jedes in der Realität beobachtete Verhalten lässt sich jedoch allein anhand von Kosten und Nutzen modellieren: Aus Sicht eines Teilnehmers sind bestimmte Verhaltensweise nicht rational erklärbar, da sie innerhalb des Systems keinerlei Nutzen generieren. Ein typisches Beispiel dafür ist die Platzierung von Spam, der aus

Sicht des Angreifers z.B. dazu dient, Aufmerksamkeit auf eine bestimmte Webpräsenz zu ziehen. Darüber hinaus ist dieses Problem nicht spezifisch für das Szenario eines P2P-Systems, wie z.B. die zum Teil massiven Denial-Of-Service-Attacken im World-Wide-Web vor einigen Jahren zeigten.

Dieses aus Systemsicht nicht rationale Verhalten entzieht sich oft den im vorangegangenen Abschnitt beschriebenen Anreizmechanismen und ist daher in der Lage, dem System schweren Schaden zuzufügen. Die einzige wirksame Gegenmaßnahme besteht darin, die Kommunikation mit dem Angreifer so schnell wie möglich zu unterbrechen und diesen zu isolieren. Zusammenfassend werden alle Verhaltensweisen, bei denen ein oder mehrere Teilnehmer unter Aufwendung eigener Ressourcen dem System Schaden zufügen, als *bösartig* bezeichnet.

Anforderungen. Zusammenfassend müssen die beabsichtigten Mechanismen die folgenden Anforderungen erfüllen:

- Free-Riding darf keine dominante Strategie sein. Ein sich nicht beteiligender Peer darf keinen Nutzen aus der Arbeit Anderer ziehen können.
- Um Angriffe bösartiger Teilnehmer abwehren zu können, muss das System in der Lage sein, Angreifer auszuschließen.
- Nicht crawlende Teilnehmer werden nicht ausgeschlossen, da sie dem System keinen Schaden zufügen.

3.2 Identifikation und Behandlung bösartiger Peers

Feedback. In Abwesenheit eines zentralen Koordinators ist die Erkennung bösartiger Teilnehmer schwierig. In einem einfacheren Szenario zeigt [FLSC04], dass die Verfügbarkeit von globalen Informationen über das Verhalten von Teilnehmern in der Vergangenheit die Erkennung bösartiger Teilnehmer vereinfacht. Durch Aggregation von Feedback kann ein *Reputationswert* für jeden Teilnehmer errechnet werden, der die Informationen über das bisherige Verhalten des Teilnehmers zusammenfasst. Die Implementierung eines solchen Reputationsverzeichnisses kann wiederum rein P2P-basiert mit Hilfe von DHTs erfolgenden, wie z.B. [AD01] zeigt.

Wir setzen voraus, dass kooperative Teilnehmer Feedback über ihre Erfahrungen mit anderen Teilnehmern geben. Experimentelle Studien wie [FG00] unterstützen diese Annahme: [FG00] zeigt, dass, selbst wenn kein unmittelbarer Nutzen erwartet werden kann, einzelne Individuen bereit sind, Free Rider zu bestrafen. In unserem Szenario sind die Kosten für die Übermittlung von Feedback extrem gering, verglichen mit den Aufwendungen für das eigentliche Crawling. Ausgehend von den Annahmen in Abschnitt 1.3 wird kooperatives Verhalten somit zu positivem Feedback führen.

Auf einem höheren Abstraktionsniveau betrachtet, besteht der Grund für die Teilnahme eines kooperativen Peers am hier beschriebenen P2P-System in der Erzielung von Nutzen, der jedoch an der Existenz des Systems gebunden ist. Ein kooperativer Peer wird also sicher keine Anstrengungen unternehmen, die Existenz des Systems zu zerstören, solange dafür keine Anreize bestehen. Feedback über andere Teilnehmer, egal ob negativ

oder positiv, hat keine Verbesserungen oder Verschlechterungen der eigenen Situation zur Folge: In unserem Entwurf wird die Reputation eines Teilnehmers niemals mit der der Anderen verglichen, es werden nur die absoluten Werte der Reputation betrachtet. Somit gibt es keinen rationalen Anreiz für falsches Feedback. Weiterhin sieht unser Ansatz nur wenige diskrete Arten von Feedback-Objekten (wie z.B. 'Anfrage (nicht) beantwortet', 'Verifikation (nicht) bestanden') vor, die jeweils ein festes Gewicht besitzen. Der Einfluss, den das Feedback eines Teilnehmers auf einen Anderen ausüben kann ist damit begrenzt. Aufgrund der Mehrheit der kooperativen Teilnehmer, die jeweils wahres Feedback abgeben, ist der Einfluss falschen Feedbacks ausgehend von einer Minderheit bösartiger Peers ebenfalls begrenzt. Mechanismen wie [KSGM03, AD01] ermöglichen es darüber hinaus, Manipulationen auf der Ebene der Reputationsspeicherung zu reduzieren.

Reputation. Grundsätzlich existieren verschiedene Ansätze, Feedback in verteilten Umgebungen zu sammeln und in möglichst manipulationssicherer Weise zu einem Reputationswert zu aggregieren [KSGM03, AD01]. Im Folgenden gehen wir von der Existenz einer solchen Technologie aus. Die Reputation eines Teilnehmers i zum Zeitpunkt t ergibt sich folgendermaßen:

$$R_t^{(i)} = min\{R_{t-1}^{(i)} + \sum_j fb_j, R_{max}\}$$

fb_j ist dabei der numerische Wert eines einzelnen Feedback-Objektes. j ist ein Iterator über die Feedback-Objekte des aktuellen Zeitraums t. Da ein Teilnehmer während einer Periode mit mehreren Anderen interagieren kann, können grundsätzlich mehr als Eines zwischen den Neuberechnungen des Reputationswertes ankommen. Alle Feedback-Inkremente werden mit gleichem Gewicht addiert.

Ausgehend von der Reputation, können bösartige Teilnehmer effektiv ausgeschlossen werden: Wir definieren dazu eine minimale Reputation R_{min}. Jede Nachricht eines Teilnehmers mit Reputation $<= R_{min}$ wird verworfen. Am anderen Ende der Skala definieren wir eine maximale Reputation R_{max}. Das implementiert das Prinzip des 'Kurzzeitgedächtnisses' und reduziert wie in [FLSC04] beschrieben den Einfluss von Teilnehmern, die nach einer Zeit der Kooperativität zu 'Verrätern' werden. Diese Mechanismen beziehen sich jedoch ausdrücklich nur auf die Reputation eines Teilnehmers, nicht auf dessen erbrachter Leistung (Beteiligung).

Umgang mit neuen Teilnehmern. Von besonderer Bedeutung ist der Umgang mit neu hinzu kommenden Teilnehmern. Im hier vorgestellten Ansatz wird ihnen eine Reputation von $R_{initial}$ zugewiesen. Diese sollte der folgenden Bedingung genügen: $R_{min} < R_{initial} \leq R_{max}$. Die Wahl der initialen Reputation stellt dabei die Einstellung des Systems gegenüber neuen Peers dar. Ein hoher Wert führt zu einem System, was jeden neuen Peer als kooperativ betrachtet, solange keine Beschwerden über ihn eingehen. Die Wahl von $R_{initial}$ hat jedoch keine Auswirkungen auf die Beurteilung der Crawlingleistung eines Teilnehmers und dessen Möglichkeit, auf Nutzdaten zuzugreifen. Die Wahl eines hohen Wertes für $R_{initial}$ ist damit möglich, ohne Free Ridern die Tür zu öffnen. Jedoch erhöht ein zu hoher Wert die Chancen eines Angreifers, Schaden im System anzurichten: Es muss mehr negatives Feedback eingehen, bis der Angreifer erkannt und ausgeschlossen wird. Im Gegensatz dazu führt die Wahl einer initialen Reputation von $R_{initial} = R_{min} + \epsilon$

zu einem sehr misstrauischen System[3]. Wir zeigen an späterer Stelle, dass eine solche Wahl nicht optimal ist, da sie eine einfache Denial-of-Service-Attacke ermöglicht: Sehr wenig (falsches) negatives Feedback genügt hier, um einen neuen Teilnehmer sofort wieder auszuschließen, selbst wenn der Teilnehmer sich kooperativ verhält. In den in Abschnitt 4.3 beschriebenen Experimenten werden wir den Einfluss der initalen Reputation daher näher quantifizieren.

3.3 Erkennung passiven Verhaltens

Die Erkennung passiver Teilnehmer ist nicht so einfach, wie das auf den ersten Blick erscheinen mag: Da jeder Teilnehmer laut Annahme seine gesammelten Daten selbst speichert, gibt es keinen allgemeinen Weg, die Aktivität der einzelnen Teilnehmer zu beobachten. Die erbrachte Leistung ist jedoch ausgesprochen wichtig, um die Berechtigung eines Teilnehmers zum Absetzen von Anfragen zu überprüfen. Die Crawlingleistung eines Teilnehmers wird als ausreichend definiert, wenn sie größer als die s-fache durchschnittliche Beteiligung aller kooperativen Peers ist. Um einen Crawlingfortschritt zu gewährleisten, genügt es, Werte von s um 1.0 zu wählen. Unter der Bedingung, dass es wenigstens einen ständig crawlenden kooperativen Peer gibt, wird damit die durchschnittliche Leistung im System stets steigen, was wiederum andere Teilnehmer zum Crawlen motiviert. Jeder Teilnehmer kann die durchschnittliche Leistung recht einfach lokal beobachten: Jeder anfragende Teilnehmer muss seine bislang erbrachte Leistung mitteilen. Experimente haben gezeigt, dass ‚ot der Zeit die einzelnen Teilnehmer ein recht zutreffendes Bild der durchschnittlichen Leistung durch gleitende Mittelwertbildung aufbauen können.

Überprüfung einer behaupteten Crawlingleistung - Algorithmus. Grundsätzlich sind Selbstauskünfte über die Leistung von Teilnehmern nicht besonders glaubwürdig: Da der mitgeteilte Wert einen direkten Einfluss auf die Zulässigkeit einer Anfrage hat, ist die Motivation, dabei zu lügen, für rationale Teilnehmer hoch. Wenn die Korrektheit einer solchen Angabe jedoch überprüfbar ist und eine falsche Angabe mit einer Strafe sanktioniert wird, ist eine solche Selbstauskunft jedoch eine gute Entscheidungsgrundlage. Im Gegensatz zu anderen Verfahren (wie z.B. quorumbasierte Ansätze, in denen mehrere Teilnehmer die gleiche Partition bearbeiten und anschließend die Ergebnisse vergleichen) skaliert diese hervorragend. Ein Challenge-Response-basiertes Verfahren, dass die behauptete Crawlingleistung eines Teilnehmers verifiziert, zeigt Abbildung 1.

Unser Verifikationsprotokoll baut dabei auf der unvorhersehbaren Struktur des Webs auf: Sofern kein komplettes Verzeichnis aller Webseiten existiert, ist das Wissen eines Peers A um eine Webseite P ein starker Hinweis darauf, dass A eine Seite mit einem Link zu P gecrawlt hat. Dazu führt jeder Peer eine nach der Domain der Website sortierte Liste aller bislang besuchten Seiten. Ein Teilnehmer B, der die von A behauptete Leistung überprüfen möchte, fordert nun eine Teilmenge von A's gecrawlten Seiten (Zeilen 4-7) an. Das kann z.B. anhand von Hashwerten des Domainnamens und der Auswahl eines Teilbereichs des Wertebereichs der Hashfunktion geschehen. Geht man von einer Gleichverteilung der Er-

[3]Das kleine Inkrement ϵ verhindert den sofortigen Ausschluss eines neu beitretenden Teilnehmers

```
1  function verifyCrawlContrib(
2     remotePeer, clContrib ) : boolean
3  begin
4     Partition rpart = remotePeer.getPart()
5     Partition spart = randomSubPart(rpart,
6                                clContrib);
7     linkData = remotePeer.getSample(spart);
8     realC = countLinks(linkData);
9
10    if (realC∗ sizeof(rpart) / sizeof(spart)
11       < clContrib) return FALSE;
12
13    return verifySample(linkData)
14 end
```

Abbildung 1: Der Verifikationsalgorithmus

gebnisse der Hashfunktion aus, ermöglicht die Zahl der von A gelieferten Ergebnisse eine Abschätzung der tatsächlich gecrawlten Seiten (Zeilen 8-11). Die Vollständigkeit des Crawlings von A ist überprüfbar, indem B einzelne Sites aus der Stichprobe erneut crawlt (Zeile 13). Letztlich kann auch eine 'semantische' Überprüfung in die Verifikation einbezogen werden, indem auch der Prozess der applikationsspezifischen Inhaltsextraktion einbezogen und Ergebnisse von B mit denen von A verglichen werden. Der Aufwand für diese Art der Verifikation hängt ausschließlich von der Größe der gewählten Subpartition ab, selbst wenn die Größe der Crawlingpartition über die einzelnen Peers stark variiert. Die Größe des Webs oder die Anzahl der Peers hat hingegen keinen Einfluss. Der Gesamtaufwand ist damit konstant, was eine gute Skalierbarkeit verspricht.

Häufigkeit der Verifikation. Wie oft sollte man nun prüfen, ob die behauptete Crawlingleistung eines Peers wahr ist? Anstatt bei jedem einzelnen Request eines Teilnehmers ein solches Verfahren durchzuführen, schlagen wir vor, die Verifikation nur mit einer gewissen Wahrscheinlichkeit P_{ver} durchzuführen, die von der Reputation des betreffenden Peers abhängig ist. Aus globaler Sicht sollte diese für Teilnehmer mit hoher Reputation gering, für Peers mit sehr niedriger Reputation dagegen 1 sein. Dieses Vorgehen konzentriert die Verifikationsmaßnahmen auf potenziell bösartige Teilnehmer und motiviert als Nebeneffekt kooperative Teilnehmer, eine hohe Reputation zu bewahren.

Wir schlagen daher folgende Heuristik vor:

$$P_{ver} = \max\left(P_0 - \frac{R - R_{min}}{R_{max} - R_{min}} + 1; 1\right)$$

P_0 ist dabei die minimale Verifikationswahrscheinlichkeit für Peers mit maximaler Reputation. Dieser Ansatz hat in Experimenten einen guten Ausgleich zwischen Aufwand und zuverlässigem Ausschluss bösartiger Teilnehmer gezeigt. In der Evaluation kommen wir auf die quantitativen Wirkungen dieser Heuristik zurück.

3.4 Anfragebearbeitung

Teilnehmer ziehen ihren Nutzen aus dem System, indem sie Anfragen an Andere absetzen können. Der genaue Inhalt ist applikationsspezifisch, im Beispiel einer P2P-Suchmaschine wird eine solche Anfrage vermutlich aus Suchworten und eventuell weiteren Attributen

(wie z.B. Zeitraum der letzten Änderung, Sprache, Größe) bestehen. Die Anfragebearbeitung besteht aus einen Sender- und einem Empfängerteil. Der Sender, der am Absetzen eines Requests interessiert ist, muss eine hinreichende Beteiligung (siehe Abschnitt 3.3 als Parameter des Requests übermitteln. Ist die Beteiligung nicht hinreichend, wird ein kooperativer Teilnehmer die Antwort verweigern. Abhängig vom Erhalt der Antwort gibt der Absender des Requests entsprechendes Feedback.

Der Empfänger des Requests vergleicht die enthaltene behauptete Crawlingleistung des Absenders mit seiner Schätzung der durchschnittlichen Leistung. Anschließend ermittelt er die Reputation des Anfragenden und startet mit einer gewissen Wahrscheinlichkeit $P_{ver}(R)$ eine Verifikation. Abhängig von deren Ergebnis übermittelt er entsprechendes Feedback. Wenn die behauptete Crawlingleistung sich als wahr herausgestellt hat und sie hinreichend ist, wird der Request beantwortet.

4 Evaluation

Abschnitt 3.1 listet die wichtigsten Typen unerwünschten Verhaltens auf. Um ein Modell als Basis für unsere Experimente zu entwickeln, haben wir typische Muster von Benutzerfehlverhalten in heutigen Websystemen betrachtet. Dabei traten zwei Hauptbedrohungen zu Tage: Free Riding und Denial-of-Service-Attacken. Die folgenden Experimente sollen zeigen, dass die bislang beschriebenen Experimente damit umgehen können.

Zu diesem Zweck benötigen wir eine Art Worst-Case-Verhalten eines maximal destruktiven Peers. Von besonderer Bedeutung ist dabei die Frage, ob ein solcher Peer crawlen sollte. Falls er das tun würde, hat ein solcher bösartiger Teilnehmer nahezu identische Kosten wie ein kooperativer, da wir davon ausgehen, dass bei einem effizienten Systementwurf die Aufwendungen für das Crawling alle Anderen bei weitem übersteigen. Zum Angriff eines großen Systems aus vielen tausend Teilnehmern muss ein Angreifer damit Ressourcen in einer ähnlichen Größenordnung wie der bereits im System vorhandenen Kapazität zur Verfügung stellen. Verzichtet ein Angreifer jedoch auf das Crawling, würde er einen erheblichen Ressourcenvorteil erzielen bzw. die vorhandenen erheblich destruktiver einsetzen können. Diese Variante erscheint uns erheblich gefährlicher, so dass wir uns in unserem Worst-Case-Modell darauf konzentrieren.

Das beabsichtigte Verhaltensmodel beruht auf der folgenden Grundidee: Ein bösartiger Peer crawlt nicht. Trotz allem behauptet er natürlich, gecrawlt zu haben. In der Folge wird er jedoch keine Chance haben, eine Verifikation der Crawlingleistung zu bestehen. Um maximalen Schaden anzurichten, gibt ein solcher Peer negatives Feedback über alle kooperativen Teilnehmer, zusätzlich 'beschützt' er andere bösartige Peers, indem er positives Feedback über sie abgibt (Verschwörungsprinzip).

4.1 Experimentierumgebung

Die Evaluierung eines realen Webcrawlers ist schwierig bis unmöglich, sofern man eine realistische Abdeckung des gesamten Webs erreichen will. Neben einem hohen Bedarf an Hardwareressourcen stellt die Durchführung eines solchen Experiments eine starke Belastung der Netzwerk-Infrastruktur dar, selbst Auswirkungen auf das gesamte Web sind denkbar. Weiterhin ist es schwierig, auf diese Weise einzelne Effekte gezielt, reproduzierbar und isoliert zu untersuchen, da sich das Web stets verändert. Wir führen unsere Untersuchungen daher anhand einer Simulationsumgebung durch, die in diskreten Zyklen arbeitet. Jedem Teilnehmer wird pro Zyklus ein Zeitschlitz zugewiesen. P2P-Webcrawling funktioniert nur, wenn andere Teilnehmer ein Interesse an den beim Crawling gesammelten Daten haben. Um dieses Interesse nachzubilden, werden Requests an zufällig ausgewählte Teilnehmer gesendet. In jedem Zyklus führt ein Peer drei grundlegende Aktivitäten durch: Zunächst wird ein Request an einen anderen Peer gesendet. Anschließend werden alle eingegangenen Requests bearbeitet und ggf. beantwortet. Dieser Schritt schließt unter Umständen die Vergabe von Feedback ein. Anschließend wird bis zum Ende des Zyklus gecrawlt. Abbildung 2 zeigt den Simulationszyklus in Pseudocode.

```
1  proc mainloop()
2  begin
3     Key requestKey = random()
4     answered = sendRequest(requestKey)
5
6     if answered submitFeedBack(fbReqAnsw)
7     else submitFeedBack(fbReqFail)
8
9     foreach request in queue do
10       handleRequest(request)
11    end
12
13    crawl()
14 end
```

Abbildung 2: Simulationsablauf

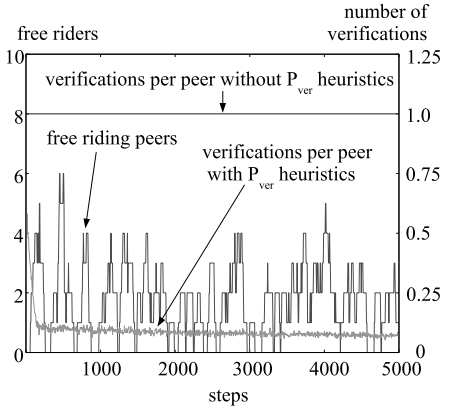

Abbildung 3: Neue Peers treten einem System aus kooperativen Teilnehmern bei

Auf dieser Implementierung aufbauend werden im Folgenden zwei Szenarien betrachtet. Das Erste wird verwendet, um ausgehend von einem eingeschwungenen System ohne bösartige Teilnehmer die Stabilität gegenüber einen beständigen Zustrom von Peers zu untersuchen, von denen ein bestimmter Teil Free Riding beabsichtigt. Dieses Szenario wurde entworfen, um den normalen Betriebszustand zu simulieren.

Das zweite Szenario untersucht die Auswirkungen von massiv auftretendem bösartigen Verhalten. Dazu wird wiederum ein eingeschwungenes System ohne bösartige Teilnehmer aufgesetzt, dem innerhalb eines Zyklus eine große Zahl bösartiger Teilnehmer beitreten. An diesem Punkt beginnt die Simulation. Während des weiteren Verlaufs treten keine neuen Peers bei.

4.2 Sporadische Free-Rider

Free Riding entsteht, wenn der Zugriff auf wertvolle Inhalte an eine Kompensation gebunden ist. Das Hauptziel eines Free-Riders ist Aufwandsminimierung, hierin besteht ein grundsätzlicher Unterschied zu bösartigen Teilnehmern. Um als kooperativ zu gelten, muss ein Peer seine wahre Crawlingleistung übermitteln. Offensichtlich ist das keine Strategie, die erfolgreiches Free-Riding ermöglicht, denn alle kooperativen Teilnehmer werden solche Requests berechtigterweise nicht beantworten. Wir gehen daher in diesem Experiment davon aus, das ein Free Rider sich nicht kooperativ verhält und eine falsche Leistungsbehauptung aufstellt. Damit riskiert er natürlich Reputationsverluste und seinen Ausschluss.

Das Experiment beginnt mit einem Szenario aus 1000 kooperativen Peers mit der initalen Reputation $R_{initial}$=0. Nach dem Start der Simulation treten neue Teilnehmer bei, einer pro Zyklus. 10% davon sind Free Rider und geben falsche Leistungen vor. Anhand der P_{ver}-Heuristik wird die Wahrscheinlichkeit der Verifikationen gesteuert. Abbildung 3 zeigt die Anzahl der Verifikationen pro Zyklus und Peer sowie die absolute Anzahl der Free-Rider im System über die Zeit. Die Zahl der Verifikationen pro Peer ohne Anwendung der P_{ver}-Heuristik ist etwa 1, da jeder Teilnehmer pro Zyklus genau einen Request absetzt. Unter Verwendung der Heuristik geht nach einer Phase häufiger Verifikationen ($t < 100$) deren Anzahl aufgrund der ansteigenden durchschnittlichen Reputation stark zurück. Im späteren Verlauf konvergiert die durchschnittliche Reputation gegen R_{max}, neu hinzu kommende Peers mit geringerer Reputation haben aufgrund der steigenden Gesamtzahl relativ betrachtet einen abnehmenden Einfluss. Die absolute Zahl der Free-Rider bleibt konstant niedrig, auch wenn ständig neue hinzu kommen. Der Verifikationsaufwand folgt der Tendenz aus dem gleichen Grund. Alles in Allem deuten diese Ergebnisse, insbesondere der sinkende Grenzaufwand der Verifikation, auf eine gute Skalierbarkeit und Robustheit gegenüber einem mit konstanter Intensität ausgetragenen Angriff hin. Die absolute Zahl der Verifikationen liegt bei ca. einer je 10 verarbeitete Requests. Die P_{ver}-Heuristik reduziert folglich in diesem Szenario die Zahl der notwendigen Verifikationen um 60-90%, ohne deren Wirksamkeit zu beeinträchtigen. Potenzielle Free-Rider, die falsche Crawlingleistungen behaupten, werden wie beabsichtigt schnell und wirksam ausgeschlossen.

4.3 Koordinierter Angriff

Die zweite Serie von Experimenten betrachtet massive koordinierte Angriffe. Anhand von bewussten Worst-Case-Annahmen sollen dabei die Grenzen unseres Ansatzes untersucht werden, auch wenn einige der im Folgenden vorgestellten Angriffe in dieser Form praktisch schwierig durchzuführen sind. Wir gehen hier grundsätzlich von einem Angreifer aus, der eine große Menge von Peers kontrolliert. Die angreifenden Peers erkennen sich untereinander, sodass sie sich gezielt gegenseitig unterstützen können (Collusion Attack). Das Ziel des Angreifers besteht darin, das System zu zerstören oder zumindest einen signifikanten Teil der kooperativen Teilnehmer aus dem System auszuschließen. Dazu versuchen die angreifenden Peers, das Reputationssystem zur Verleumdung der kooperativen Peers zu missbrauchen. Anhand von negativem Feedback über kooperative Teilnehmer

versuchen sie, deren Ausschluss zu erreichen, durch positives Feedback über andere bösartige Teilnehmer versuchen sie diese zu beschützen bzw. deren Ausschluss hinauszuzögern.

Das Szenario für dieses Experiment besteht zunächst aus 1000 Teilnehmern mit einer initialen Reputation von 0. Ein bestimmter Anteil davon ist bösartig. Während dieser variiert wird, bleiben alle anderen Parameter konstant. Abbildung 4 stellt die Zahl der kooperativen (obere Flächen) und der bösartigen Teilnehmer (untere Fläche) während 500 Zyklen dar. Während der ersten Runden ($steps < 100$) nimmt die Zahl der bösartigen Teilnehmer stark ab, zumindest für hohe Anteil von ihnen. Trotz allem kommt es zumindest bei hohen Anteilen von bösartigen Teilnehmern auch zu einigen Ausschlüssen von kooperativen Teilnehmern. Wenig später stabilisiert sich das System, alle bösartigen Teilnehmer sind ausgeschlossen, weitere Verluste von kooperativen Teilnehmern treten nicht auf. Das Resultat zeigt, das ein Angreifer mindestens 10% aller Teilnehmer stellen muss, um einen wahrnehmbaren Schaden im System hervorzurufen. Bei diesem Anteil gelingt es ihm, ca. 7.4% aller kooperativen Peers durch Manipulation des Reputationssystems auszuschließen. Die Verluste steigen jedoch, wenn der Anteil der bösartigen Teilnehmer größer wird. Ein Angreifer, der 20% aller Teilnehmer kontrolliert, kann damit 29% aller kooperativen Peers treffen. Bei ca. 30% könnte es ihm gelingen, die Hälfte der kooperativen Peers auszuschließen. In der Realität erscheint ein derart extremes Szenario unwahrscheinlich, so dass für die folgenden Untersuchungen der Anteil der unkooperativen Teilnehmer auf 10% festgelegt wird.

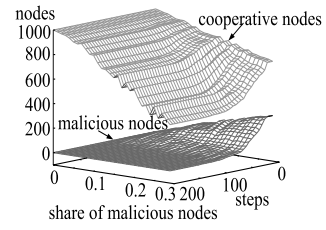

Abbildung 4: Variation des bösartigen Anteils

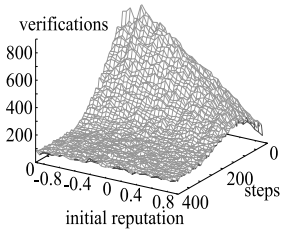

Abbildung 5: Anzahl der Verifikationen vs. $R_{initial}$

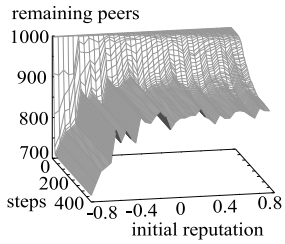

Abbildung 6: Peeranzahl vs. $R_{initial}$

Als Nächstes betrachten wir die Auswirkungen der Wahl der initialen Reputation. Im Gegensatz zum letzten Experiment wird dabei der Anteil der bösartigen Teilnehmer fix auf 10% gesetzt, die initiale Reputation variiert dagegen. Abbildung 5 stellt die Zahl der Verifikationen als Funktion der initialen Reputation dar. In den ersten Zyklen wird die Verifikationsanzahl stark durch P_{ver} bestimmt. (vgl. Abschnitt 3.3). Die Dauer bis zur Erkennung bösartiger Teilnehmer beeinflusst die weitere Entwicklung der Verifikationskosten. Da die Anzahl der bösartigen Teilnehmer für kleine initiale Reputationen schneller abnimmt, nimmt die Zahl der Verifikationen hier ebenfalls schneller ab. Die absolute Anzahl ist minimal für hohe Werte von $R_{initial}$.

Die initiale Reputation hat jedoch wiederum Auswirkungen auf das bereits betrachtete wichtige Problem der kooperativen Verluste. Abbildung 6 zeigt folgenden Zusammenhang: Eine abnehmende initiale Reputation führt zu einem monotonen und stärker werdenden Zunahme der kooperativen Verluste. Eine niedrige initiale Reputation, die man

als starkes anfängliches Misstrauen interpretieren kann, erlaubt den schnellen Ausschluss von vielen Teilnehmern - darunter auch kooperative. Aus Sicht dieses Experiments sollte $R_{initial}$ so hoch wie möglich gewählt werden. Jedoch ist auch diese Wahl nicht optimal, da sie gleichzeitig bösartigen Teilnehmern ermöglicht, länger im System zu verbleiben und somit die Langzeitstabilität im Fall eines kontinuierlichen Zustroms (vgl. erstes Experiment) gefährden kann. Aufgrund des flachen Gradienten im Bereich von -0.2 bis 1 erscheint ein Wert im Intervall $(-0.2, 0)$ ein guter Kompromiss zu sein. Der in den anderen Experimenten verwendete Wert von $R_{initial} = 0$ liegt in diesem Bereich.

5 Schlussfolgerungen

Webcrawling ist grundlegend für viele Informationssysteme. Das Anliegen dieses Artikels bestand im Entwurf eines Systems, dass das P2P-Paradigma zu Grunde legt. Damit ergeben sich viele Vorteile, aber auch einige Herausforderungen. Ohne sorgfältig entworfene Gegenmaßnahmen können Probleme wie Free Riding oder bösartiges Verhalten den Nutzen eines solchen Systems stark beeinträchtigen. Im Rahmen dieses Artikels wurden Mechanismen gegen solche unerwünschten Verhaltensweisen vorgestellt. Diese Mechanismen sind dabei gegen zwei Formen unerwünschten Verhaltens gleichermaßen wirksam: einerseits mangelnde Beteiligung, andererseits gezielt bösartiges Verhalten. Ersteres fügt dem Gesamtsystem zwar keinen Schaden zu, macht es bei massivem Auftreten aber jedoch unattraktiv für potentielle Teilnehmer. Letzteres in jedem Fall existenzbedrohend, insbesondere wenn es zu koordinierten Angriffen (collusion attack) kommt. Umfangreiche Experimente zeigten, dass unser Entwurf mit beiden Bedrohungen umgehen kann. Die Rückwirkungen der Mechanismen auf kooperative Teilnehmer sind gering, solange die Anzahl der Angreifer im Verhältnis zur gesamten Teilnehmerzahl nicht extrem hoch ist.

Literatur

[AD01] Karl Aberer and Zoran Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM01)*, 2001.

[AH00] Eytan Adar and Bernardo A. Huberman. Free Riding on Gnutella. *Technical report, Xerox PARC*, 2000.

[BB06] Klemens Böhm and Erik Buchmann. Free riding-aware forwarding in Content-Addressable Networks. *International Journal on Very Large Data Bases (VLDB)*, 2006.

[BCSV04] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. UbiCrawler: a scalable fully distributed web crawler. *Softw. Pract. Exper.*, 34(8), 2004.

[CGM02] J. Cho and H. Garcia-Molina. Parallel Crawlers. In *Proc. of the 11th International World–Wide Web Conference*, 2002.

[Chr05] Michael Christen. YaCy - P2P-based distributed Web Search Engine. http://www.yacy.net/yacy, 2005.

[FG00] Ernst Fehr and Simon Gaechter. Cooperation and Punishment in Public Goods Experiments. IEW - Working Papers iewwp010, Institute for Empirical Research in Economics, 2000.

[FLSC04] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, New York, NY, USA, 2004.

[GS05] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, New York, NY, USA, 2005.

[HN99] Allan Heydon and Marc Najork. Mercator: A Scalable, Extensible Web Crawler. *World Wide Web*, 2(4), 1999.

[KSGM03] Sepandar Kamvar, Mario Schlosser, and Hector Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, New York, NY, USA, 2003.

[LFSC03] K. Lai, M. Feldman, I. Stoica, and J. Chuang. Incentives for cooperation in peer-to-peer networks. In *In Workshop on Economics of Peer-toPeer Systems*, 2003.

[LNS93] Witold Litwin, Marie-Anne Neimat, and Donovan A. Schneider. LH: Linear Hashing for distributed files. In *Proceedings of the 1993 ACM SIGMOD international conference on Man agement of data*, 1993.

[R+01] Sylvia Ratnasamy et al. A scalable content-addressable network. In *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

[S+01] Ion Stoica et al. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

[S+03] Torsten Suel et al. ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. In *WebDB*, 2003.