Diplomarbeit

Gossiping

Dirk Maier

Betreuung: Ruedi Arnold
Leitung: Prof. Dr. Roger Wattenhofer

Distributed Computing Group
Departement Informatik
ETH Zürich

Wintersemester 2003/2004

Zusammenfassung

Gossiping ist ein Verfahren, um in einem Netzwerk eine Information an alle an einem Netzwerk teilnehmenden Prozesse zu verschicken. Hierzu werden Nachrichten an zufällig ermittelte Adressaten gesandt.

In dieser Diplomarbeit wird das Prinzip von Gossiping vorgestellt und die verschiedenen möglichen Konzepte und Vorgehensweisen besprochen. Es wird gezeigt, wie man Gossiping in einem Peer-to-Peer Netzwerk umsetzten kann. Für Gossiping in Peer-to-Peer wird ein konkreter Algorithmus vorgeschlagen und analysiert. Für diesen Algorithmus wird bewiesen, dass er mit hoher Wahrscheinlichkeit alle Knoten des Netzwerks informiert.

Zuletzt wird Gossiping mit anderen Broadcast-Algorithmen verglichen. Dabei stellt sich heraus, dass Fluten in Bezug auf Nachrichten- und Zeitaufwand Gossiping ebenbürtig ist und sich auch nicht anfälliger gegenüber Ausfällen im Netzwerk zeigt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Netzwerke

Es gibt in der heutigen Zeit wohl kaum noch Computer, die nicht vernetzt sind. Die meisten Rechner sind in irgendeiner Form Teil eines grösseren Netzwerks. Netzwerke ermöglichen es, teure Ressourcen miteinander zu teilen und schnell und unkompliziert Informationen auszutauschen. Besonders das Internet bietet sich als Netz der Netze an, um Informationen und auch mehr auszutauschen.

Die rasante technische Entwicklung macht Netzwerkverbindungen immer billiger und schneller. *ADSL* und ähnliche Technologien ermöglichen jeder Privatperson einen Hochgeschwindigkeits-Zugang zum Internet. Dadurch wird es für jeden möglich, schnell grosse Mengen an Daten zu transportieren. Durch die steigende Verbreitung von *wireless LAN*, also kabellosen Netzwerken, wird es auch mobilen Anwendern möglich, sich schnell und unkompliziert zu vernetzen.

Es ist durchaus denkbar, dass in naher Zukunft auch Alltagsgegenstände wie Fernseher und Kühlschränke oder gar Toaster und Waffeleisen in irgendeiner Form miteinander vernetzt sind.

1.1.1 Broadcast

Sobald sich mehrere Computer zu einem Netzwerk zusammenschliessen, wird die Kommunikation untereinander wichtig. Ein Rechner muss die Möglichkeit haben, mit den anderen Informationen auszutauschen, sonst wird das Netzwerk schnell nutzlos.

In einem Broadcast Szenario möchte ein einzelner Teilnehmer des Netzwerks eine ihm bekannte Information dem gesamten Netzwerk mitteilen. Dies soll möglichst schnell und unkompliziert möglich sein und das Netzwerk nicht unnötig belasten. Auf der anderen Seite muss in einem Netzwerk immer damit gerechnet werden, dass einzelne Nachrichtenpakete verloren gehen oder Rechner ausfallen. Ein geeignetes Verfahren zur Verbreitung der Information sollte also möglichst stabil funktionieren. Das in dieser Arbeit vorgestellte Gossiping ist eine Möglichkeit eines stabilen, einfachen Broadcast-Verfahrens.

1.1.2 Peer-to-Peer Systeme

Im Gegensatz zu einem Client/Server Netzwerk herrscht in einer Peer-to-Peer (P2P) Umgebung totale Gleichberechtigung. Jeder Teilnehmer des Netzwerks ist den anderen gleichgestellt und übernimmt gleichzeitig sowohl die Rolle eines Servers als auch die eines Clients. Ein P2P-Netzwerk ist dezentral und symmetrisch aufgebaut.

Der Allgemeinheit bekannt wurde P2P hauptsächlich durch viel diskutierte Musik- und Videotauschbörsen wie *Napster* [3] oder *KaZaa* [2]. Auch wenn P2P mit Anwendungen im juristischen Graubereich grossen Bekanntheitsgrad erlangt hat, gibt es durchaus auch legale Nutzungsgebiete, wie z.B. das *Intel® Philanthropic Peer-to-Peer Program* [1], welches P2P benützt, um die Rechenleistung von vielen Desktop PCs zu bündeln.

Da ein P2P System keinen zentralen Server hat, der die Verteilung koordiniert, ist ein Broadcast keine triviale Aufgabe. In der vorliegenden Arbeit wird der Ansatz des Gossiping auf P2P Systeme angewandt und untersucht.

1.2 Epidemische Vorgänge

Epidemische Vorgänge treten in der Natur vor allem bei Krankheiten auf. Ist ein Erreger ansteckend genug, sind in kurzer Zeit grosse Teile einer Population infiziert. Ein infiziertes Individuum wird dann zum Träger des Erregers und steckt weitere Individuen an. Dies so lange, bis das erkrankte Individuum entweder stirbt oder gesundet.

Um Epidemien sinnvoll bekämpfen zu können, wurden verschiedene mathematische Modelle und Berechnungsgrundlagen entwickelt. Mit Hilfe dieser Modelle kann abgeschätzt werden, wie schnell sich ein Infekt ausbreitet und welche Teile einer Population davon betroffen sind.

Die Idee von Gossiping ist es, epidemische Vorgänge zur Verbreitung einer Information zu nutzen. Dabei wird die zu verteilende Information als Infekt betrachtet und die an ein Netzwerk angeschlossenen Prozesse als Individuen einer Population. Rechner können sich nun gegenseitig infizieren, indem sie die Information enthaltende Nachrichten schicken.

1.3 Aufgabenstellung dieser Diplomarbeit

Zuerst stand die Einarbeitung in das Thema und Literaturrecherche auf dem Programm. Verschiedene Gossip-Protokolle sollten untersucht werden.

Als erster theoretische Schwerpunkt war dann die Umsetzung von Gossiping in einem asynchronen Netzwerkmodell vorgesehen. Zu erforschen war die Anzahl Nachrichten, die verschickt werden muss, um mit hoher Wahrscheinlichkeit das gesamte Netzwerk zu erreichen.

Der zweite theoretische Schwerpunkt der Aufgabe war die Betrachtung des dynamischen Falls. Zu untersuchen waren die Probleme die sich durch häufiges Kommen und Gehen von Knoten ergeben.

Als praktischer Teil der Aufgabe war die Implementierung eines oder mehrerer Gossip-Algorithmen auf dem vorgegebenen Forschungsprototypen *Clippee* [10] vorgesehen. Dies wurde jedoch nicht ausgeführt.

1.4 Ablauf der Diplomarbeit

Zu Beginn der vorliegenden Diplomarbeit stand eine Aufgabenstellung in der die Schwerpunkte der Arbeit festgelegt wurden. Die Beschreibung der Aufgaben war hierbei flexibel und dynamisch gedacht, so dass der Verlauf der Arbeit den erworbenen Erkenntnissen angepasst werden konnte. Recht schnell wurde klar, dass das Hauptaugenmerk auf Gossiping in einer typischen P2P Umgebung gerichtet sein würde. Es wurde versucht, einen praktikablen Algorithmus zu entwickeln, welcher das Broadcast Problem in einem P2P Netzwerk stabil und unkompliziert zu lösen vermag. Nach der Formulierung der Idee wurde dieses vorgeschlagene Verfahren dann mathematisch analysiert und die Wahrscheinlichkeit einer erfolgreichen Informationsverteilung abgeschätzt.

Da im Verlauf der Arbeit klar wurde, dass der vorgeschlagene Gossiping Ansatz für ein Peer-to-Peer System in der Praxis nur wenig sinnvoll ist, wurde die praktische Umsetzung nicht weiter verfolgt.

1.5 Kapitelübersicht

Kapitel 1 gibt eine kurze Einführung in die vorliegende Diplomarbeit. Der Ablauf der Arbeit wird in einem kurzen Abriss vorgestellt.

Kapitel 2 erklärt das grundlegende Broadcast Problem, das in dieser Arbeit untersucht wird. Es wird erklärt welche Eigenschaften ein guter Broadcastalgorithmus haben sollte.

Kapitel 3 führt den Begriff des Gossiping ein. Es wird auf den biologischen Hintergrund des Verfahrens eingegangen und verschiedene Konzepte des Gossiping werden vorgestellt. Ein für Gossiping praktikables Modell, das SIR-Modell, wird näher betrachtet. Es wird gezeigt welche mathematischen Erkenntnisse über den Verlauf des Verteilvorgangs gezogen werden können. Abschliessend werden zwei bereits bekannte Gossiping Algorithmen vorgestellt.

Kapitel 4 setzt den Ansatz des Gossiping in eine P2P Umgebung. Zunächst wird ein kurzer Abriss über P2P Systeme gegeben. Anschliessend wird betrachtet, welche Probleme sich für einen Gossiping-Algorithmus in einer solchen Umgebung stellen. Es werden zwei verschiedene Möglichkeiten zur Umsetzung von Gossiping in P2P vorgestellt.

Kapitel 5 stellt einen, an die Gegebenheiten eines P2P Netzwerks angepassten, Algorithmus vor. Es wird motiviert warum dieser Ansatz gegenüber herkömmlichen Gossiping-Algorithmen in einer P2P-Umgebung Vorteile hat. Hauptteil dieses Kapitels ist der Beweis, dass der Algorithmus mit hoher Wahrscheinlichkeit sämtliche Knoten des Netzwerks informiert.

Kapitel 6 vergleicht verschiedene Broadcast-Algorithmen. Verschiedene Gossipingansätze werden mit anderen Verfahren verglichen und verschiedene Stärken und Schwächen besprochen.

Kapitel 7 zieht ein Fazit der Arbeit und diskutiert die Ergebnisse und gibt einen kurzen Ausblick.

Kapitel 2

Broadcast Problem

Computer werden zu Netzwerken zusammengeschlossen um gemeinsame Ressourcen zu nutzen, Daten auszutauschen oder auch um einfach miteinander kommunizieren zu können. Ein Netz von verschiedenen Computern kann nahezu beliebig gross werden. Ein gutes Beispiel hierfür ist das Internet, mit dem die meisten Computer heute verbunden sind.

Ein Netzwerk kann als Graph betrachtet werden. Die Kanten entsprechen hierbei den Verbindungen zwischen den einzelnen Computern. Die Computer selbst werden als Knoten dargestellt. Einzelne Teilnehmer des Netzwerks werden fortan in dieser Arbeit schlicht Knoten genannt. Die Anzahl der Knoten im Netz wird durchgehend mit n bezeichnet.

Je nach Beschaffenheit eines Netzwerks kann nicht von einem stabilen Zustand ausgegangen werden. Sich neu an das Netz anschliessende oder aus dem Netz austretende Knoten sorgen für eine sich ständig ändernde Topologie.

Das Broadcast Problem ergibt sich aus der Situation, dass ein Knoten eine Information hat, die er sämtlichen anderen Knoten im Netzwerk mitteilen will. Eine solche Information kann zum Beispiel eine Veränderung von gemeinsam genutzten Daten sein. Abbildung 2.1 illustriert die Ausgangssituation und das Ziel.

Es sind verschiedene Ansätze bekannt, um dies zu erreichen. Im Folgenden sollen zunächst die Eigenschaften aufgelistet werden, die ein Broadcast Algorithmus haben sollte.

Korrektheit: Ein Broadcast Algorithmus muss natürlich alle Knoten erreichen. Eine hundert prozentige Korrektheit kann jedoch insbesondere bei probabilistischen Verfahren nur schwer oder gar nicht garantiert werden. Deshalb begnügt man sich in solchen Fällen damit, zu zeigen, dass *mit hoher Wahrscheinlichkeit* [1] alle Knoten erreicht werden.

Terminierung: Nachdem alle Knoten informiert wurden, muss der Algorithmus terminieren. Idealerweise sollte der Algorithmus genau dann terminieren, wenn der letzte Knoten informiert ist.

Robustheit: Oft kann nicht garantiert werden, dass das Netzwerk, in dem die Information verbreitet werden soll, stabil ist. Das heisst ein Broadcast

[1] Der Ausdruck *„mit hoher Wahrscheinlichkeit“* bedeutet mit einer Wahrscheinlichkeit von mindestens $1 - O(n^{-\alpha})$, für eine positive Konstante α.

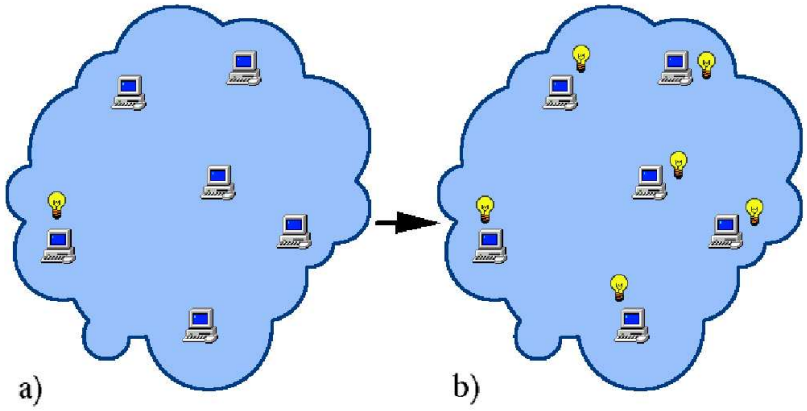

Abbildung 2.1: a) Die Ausgangssituation. Ein Knoten des Netzwerks hat eine Information, die er im Netzwerk verteilen will. b) Das Ziel. Jeder Knoten des Netzwerks kennt nun die Information.

Algorithmus für ein instabiles Netzwerk muss auch dann noch funktionieren, wenn neue Knoten dem Netzwerk beitreten oder dem Netzwerk angeschlossene Knoten dieses verlassen. Im schlimmsten Fall verlässt ein Knoten unerwartet das Netzwerk, nachdem er informiert wurde, aber selber die Nachricht noch nicht weitergeben konnte. Auch können bei der Übertragung Nachrichten verloren gehen. Diese Ausfälle müssen dann entsprechend kompensiert werden.

Skalierbarkeit: Ein Netzwerk kann potentiell alle existierenden Computer umfassen. Auch in grossen Netzen soll mit praktikablem Aufwand ein Broadcast durchgeführt werden können. Der Nachrichten- und Zeitaufwand des Algorithmus sollte also nur langsam mit der Anzahl der im Netzwerk enthaltenen Knoten wachsen.

Ein naiver Ansatz, eine Information an alle Knoten des Netzwerks zu verteilen, ist der so genannte „one-to-all“ Broadcast. Hierbei schickt der erste informierte Knoten die Information an alle anderen Knoten des Netzwerks. Bei diesem Verfahren werden alle Nachrichten vom gleichen Knoten versandt, was zu einer grossen Belastung desselben führen kann. Der Ausgangsknoten muss zudem eine komplette Liste aller Knoten des Netzwerks haben.

Die Nachrichten können aber auch entlang eines Spannbaums geschickt werden. Hierfür wird ein Subnetz des Netzwerks berechnet, welches keine Zyklen beinhaltet. Entlang dieser Netzwerkstruktur wird dann die Nachricht weitergeleitet. Die Last wird hier auf alle Knoten verteilt. Abhängig von der Struktur des Spannbaums ist die Lastverteilung mehr oder weniger gleichmässig. Der Nachrichtenaufwand ist optimal. Allerdings muss für das Gelingen dieses Algorithmus ein Spannbaum berechnet werden, was mitunter ein schweres Problem sein kann.

Eine weitere Möglichkeit die Information zu verbreiten, stellt das so genannte

Fluten dar. Jeder Knoten schickt die Nachricht an alle seine Nachbarknoten weiter. Es werden mehr Nachrichten als benötigt verschickt, ein Knoten erhält die Information also mehrfach. Dies ist nicht unbedingt ein Nachteil, da dadurch eine gewisse Stabilität erreicht wird. In Kapitel 6 werden die verschiedenen Algorithmen näher erklärt und miteinander verglichen.

In dieser Arbeit wird mit Gossiping ein Verfahren vorgestellt und untersucht, mit dem die Nachricht zufällig verteilt wird. Jeder Knoten, der die Nachricht erhält, schickt sie an eine gewisse Menge andere Knoten weiter. Diese Kommunikationspartner werden zufällig aus allen Knoten des Netzwerks ermittelt. Um zu garantieren, dass mit hoher Wahrscheinlichkeit alle Knoten informiert werden, muss die Nachricht an genügend Knoten weiter geschickt werden. Wieviele Nachrichten ein Knoten verschicken muss, wird in den folgenden Kapiteln besprochen.

Kapitel 3

Gossiping Allgemein

Gossiping ist ein Verfahren, um eine Information probabilistisch an die Knoten eines Netzwerks zu verteilen. Ein Knoten, welcher die Information schon kennt, schickt diese an andere Knoten weiter, welche zufällig aus dem Netzwerk gewählt werden.

In diesem Kapitel wird die grundlegende Idee von Gossiping vorgestellt. Es wird beschrieben, welche Idee hinter Gossiping steckt und welche Probleme sich bei der Entwicklung eines konkreten Gossiping Algorithmus stellen. Zuletzt werden verschiedene Konzepte beschrieben und zwei bekannte Algorithmen vorgestellt.

3.1 Allgemeines

Gossiping Algorithmen werden auch „epidemic algorithms“ (epidemische Algorithmen) oder „rumor mongering“ (Gerüchtemacherei) genannt. Die zu verteilende Information wird entsprechend auch oft als „rumor“ (Gerücht) bezeichnet. Wie die Beschreibung als epidemische Algorithmen vermuten lässt, ähnelt der Vorgang eines Gossiping Algorithmus dem einer Epidemie im biologischen Sinn. Solche Epidemien sind zum Beispiel die Verbreitung von ansteckenden Krankheiten oder Virusinfektionen. Allerdings sind die Ziele in der Informatik und der Biologie grundverschieden. Ein Gossiping Algorithmus versucht so schnell wie möglich alle Knoten zu erreichen, während in der Biologie vorwiegend untersucht wird, wie man Epidemien möglichst rasch eindämmen kann. Auch hat man bei der Entwicklung eines Gossiping Algorithmus die Freiheit, die epidemischen Mechanismen seinen Bedürfnissen anzupassen.

Nichtsdestotrotz sind die Resultate der theoretischen Untersuchung von Epidemien [6] für das Verständnis und die Analyse von Gossiping sehr hilfreich. Es gibt zwei grundlegende Modelle einer Epidemie, namentlich „infect forever“ und „infect and die“.

infect forever: Dieses Modell beschreibt eine Infektion, deren Ausgang nicht tödlich ist. Nachdem ein Individuum kontaminiert wurde beginnt es damit, andere Individuen an zustecken. Dieser Vorgang geht potentiell so lange weiter, bis alle Individuen angesteckt sind.

infect and die: Wenn in diesem Modell ein Individuum angesteckt wird, verbreitet es den Infekt nur während einem gewissen Zeitraum, bevor es aufhört. Danach ist es entweder tot oder geheilt und immun gegenüber dem Infekt. Beides hat zur Folge, dass das entsprechende Individuum für die weitere Verteilung der Infektion keine Rolle mehr spielt.

Ein Gossiping Verfahren wendet die Prinzipien einer Epidemie an, um in einem Netzwerk eine Information zu verbreiten. Ein bereits informierter, also infizierter, Knoten steckt weitere, zufällig ermittelte Knoten an, indem er ihnen die Information schickt. Auf diese Weise neu informierte Knoten verfahren dann nach dem gleichen Schema. Nachdem ein Knoten genug Nachrichten versandt hat, ist er gewissermasse geheilt, also nicht mehr infektiös und hört dementsprechend auf, weitere Nachrichten zu versenden. Auf diese Art wird eine Information, die man im Netzwerk bekannt machen will, ähnlich wie ein Infekt oder eine Krankheit von Knoten zu Knoten übertragen. Die Übertragung geschieht zufällig.

Will man diesen Ansatz in einem Algorithmus umsetzen, stellen sich verschiedene Fragen. So muss entschieden werden, wieviele Nachrichten ein Knoten verschickt, bevor er inaktiv wird. Einerseits möchte man genug Nachrichten verschicken, dass mit hoher Wahrscheinlichkeit alle Knoten des Netzwerks erreicht werden, andererseits soll die Anzahl verschickter Nachrichten möglichst gering sein. Es muss auch geklärt werden, nach welchen Kriterien ein Knoten entscheidet, wann er genügend Nachrichten versendet hat. Ebenfalls muss festgelegt werden, nach welchem Verfahren die Empfänger der versandten Nachrichten ermittelt werden. Abhängig von der Netzwerktopologie kann dies eine nicht triviale Aufgabe darstellen.

3.2 Konzepte

Es gibt verschiedene Möglichkeiten einen Gossiping Algorithmus zu konzipieren. Einige grundlegende Varianten werden in [4] vorgestellt.

Zunächst kann unterschieden werden, ob der Sender der Nachricht vom Empfänger eine Antwort erhält oder nicht.

Blind: Der informierende Knoten erfährt nicht, ob der Empfänger einer Nachricht bereits zuvor informiert war oder nicht. Das heisst, ein Knoten entscheidet alleine aufgrund interner Zustände, wann er genug Nachrichten verschickt hat.

Feedback: Ein infektiöser Knoten wird von den Empfängern seiner Nachrichten über deren Zustand informiert. Er erfährt also, ob er mit einer Nachricht einen bis dahin uninformierten Knoten erreicht hat oder ob die Nachricht unnötig verschickt wurde.

Ein auf diese Weise informierter Knoten verliert das Interesse an einer Weiterverteilung nur, wenn der Empfänger der Nachricht bereits zuvor informiert war.

Nachdem ein Knoten eine Nachricht verschickt hat (Blind) oder nachdem eine Nachricht an einen bereits informierten Knoten geschickt wurde (Feedback),

muss ein infektiöser Knoten also entscheiden, ob er weitere Nachrichten verschickt oder nicht. Hierzu wird ein interner Parameter k verwendet, welcher vom Algorithmus vorgegeben wird. Auf Grund von k entscheidet ein Knoten dann, ob er eine Nachricht weiterhin verteilt, oder nicht.

Coin: Mit einer Wahrscheinlichkeit $\frac{1}{k}$ verschickt der betreffende Knoten keine weiteren Nachrichten. Das Abbruchkriterium ist also probabilistisch. Die Wahrscheinlichkeit bleibt im Verlaufe des Algorithmus gleich.

Counter: Hierbei wird k als Zähler verwendet. Jeder Knoten dekrementiert seinen Zähler k fortwährend. Das Abbruchkriterium ist deterministisch. Wird diese Strategie zusammen mit Blind verwendet, dann verschickt jeder informierte Knoten genau k Nachrichten, bevor er aufhört zu senden.

In einem Gossiping Algorithmus nimmt ein Knoten mit einem zufälligen Kommunikationspartner Kontakt auf. Man kann sagen er ruft einen zufälligen Knoten an. Zwischen diesen zwei Knoten wird dann das Gerücht kommuniziert. Dies kann in zwei Richtungen geschehen.

Push: Hierbei informiert lediglich der anrufende Knoten den angerufenen über allfällige Gerüchte.

Pull: Wenn der angerufene Knoten dem Anrufer seine Gerüchte mitteilt, spricht man von Pull-Kommunikation.

Es ist durchaus möglich sowohl Push als auch Pull anzuwenden und beide Vorgehen in einem Algorithmus anzuwenden.

3.3 SIR Modell

Das SIR Modell ist ein fundamentales Modell für die mathematische Untersuchung von der Verbreitung von Krankheiten. Da aus diesem Modell einige Erkenntnisse über Gossiping Verfahren gewonnen werden kann, wird es hier kurz vorgestellt.

Das SIR Modell oder susceptible/infective/recovered Modell wurde bereits 1920 von Lowell Reed und Wade Hampton Frost formuliert, jedoch nicht publiziert. In diesem Modell wird eine Population in drei Klassen unterteilt. Jedes Individuum wird entsprechend seinem Stand im Verlauf der Krankheit einen der Zustände *S, I* oder *R* zugeordnet.

S: Zuerst ist ein Individuum *susceptible*, also empfänglich oder anfällig. Das bedeutet, dass das Individuum von der Krankheit noch nicht angesteckt wurde, sich aber noch infizieren kann.

I: Nachdem ein Individuum angesteckt wurde, ist es *infective*, also infektiös oder ansteckend. Es ist krank und kann den Infekt auf andere übertragen. Jedes infektiöse Individuum bleibt für eine bestimmte Zeitintervall in diesem Zustand, bevor es in den Zustand R übergeht.

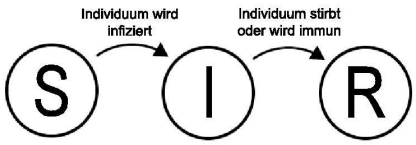

Abbildung 3.1: Übergangsdiagramm des SIR Modells

R: Zuletzt ist das Individuum *recovered* oder *removed*, also gesundet oder beseitigt. Je nach Interpretation hat es sich von der Krankheit erholt und ist gegen den Infekt nun immun, oder die Krankheit verlief tödlich und das Individuum ist verstorben. In beiden Fällen ist das Individuum nicht mehr krank und kann den Infekt nicht mehr auf andere übertragen. Auch kann es selbst nicht mehr angesteckt werden. Für die weitere Verteilung der Krankheit spielt es also keine Rolle mehr.

Zustand R ist final. Ist ein Individuum einmal in diesem Zustand, so bleibt es auch darin.

Im Verlauf der Krankheit gibt es eine feste Wahrscheinlichkeit pro Zeitschritt, dass ein infektiöses Individuum ein empfängliches, mit dem es Kontakt hat, infiziert. Individuen, welche die Krankheit tragen, bleiben für eine gewisse Zeitperiode infektiös, bevor sie sich erholen und ihre Ansteckungsfähigkeit verlieren. In einem SIR Modell wird oft von einer so genannten *fully mixed* Approximation ausgegangen. Das bedeutet, die Wahrscheinlichkeit, dass zwei Individuen miteinander Kontakt haben, ist für jedes Individuen-Paar gleich gross [6]. Zwar ist die Annahme unrealistisch, aber mathematisch relativ einfach und besonders für die Untersuchung von Gossiping praktikabel.

Es wurde gezeigt [7], dass die *fully mixed* Version des SIR Modells äquivalent zu einer Krankheitsverbreitung auf einem Erdös-Rényi Zufallsgraphen ist, wenn sich jeweils zwei Knoten des Graphen genau dann anstecken, wenn sie durch eine Kante miteinander verbunden sind. In einem Erdös-Rényi Zufallsgraphen sind alle Knotenpaare mit gleicher Wahrscheinlichkeit miteinander verbunden. Dieser Umstand kann dazu benützt werden, um mit Hilfe von Erkenntnissen aus der Graphentheorie Gossiping Algorithmen zu untersuchen. In dieser Diplomarbeit wurde dieser Ansatz jedoch nicht weiter verfolgt.

Im SIR Modell kann ein Individuum, nachdem es sich erholt hat, nicht erneut infiziert werden. Um aber solche Vorgänge beschreiben zu können, wurde das SIS Modell formuliert. In diesem Modell geht ein Individuum vom Zustand I wieder in den Zustand S über und kann dann erneut infiziert werden. Für Gossiping ist dieses Modell jedoch uninteressant und wird hier nur der Vollständigkeit halber erwähnt.

Im Folgenden werden mit Hilfe des SIR Modells zwei Verteilungsstrategien für Gossiping-Algorithmen untersucht.

3.3.1 Blind Counter

Im Laufe des Blind/Counter Algorithmus benachrichtigt jeder informierte Knoten k andere Knoten. Danach hört er auf, die Information zu verbreiten. Die von

ihm benachrichtigen Knoten geben ihm kein Feedback. Er weiss also nicht, ob diese schon zuvor informiert waren oder nicht. Im Folgenden bezeichnet in Anlehnung an das SIR Modell s den Bruchteil der uninformierten, empfänglichen Knoten, i bezeichnet den Bruchteil der infektiösen Knoten und r den Bruchteil der informierten Knoten, welche die Verteilung der Information eingestellt haben. Es gilt hierbei $s + i + r = 1$.

Wenn davon ausgegangen wird, dass jeder Knoten seine k Nachrichten gleichzeitig abschickt, können für den Verlauf von i und s folgende Gleichungen aufgestellt werden.

$$\frac{ds}{dt} = -kis \tag{3.1}$$

$$\frac{di}{dt} = kis - i \tag{3.2}$$

Um dt aus den Gleichungen (3.1) und (3.2) zu eliminieren wird eine Division durchgeführt. Dadurch erhält man $\frac{di}{ds}$.

$$\frac{di}{ds} = -\frac{kis - i}{kis} = \frac{1}{ks} - 1$$

Wenn man nun $\frac{di}{ds}$ nach s integriert, erhält man $i(s)$.

$$i(s) = \int \frac{1}{ks} - 1 ds = \frac{1}{k} \ln s - s + c$$

Um die Integrationskonstante c zu berechnen wird der Anfangszustand betrachtet. Zu Beginn ist genau ein Knoten informiert und infektiös, alle anderen Knoten sind uninformiert. Es gilt für die erste Runde $i(1 - \epsilon) = \epsilon$, wobei für grosse n ϵ gegen Null geht. Für die Konstante c ergibt dies:

$$c = s - \frac{1}{k} \ln s = 1$$

Folglich gilt für $i(s)$:

$$i(s) = \frac{1}{k} \ln s - s + 1$$

Am Ende des Verteilvorgangs wird i Null. Für s_{end}, den Anteil uninformierter Knoten nach Beendigung des Algorithmus, ergibt sich dann folgende implizite Gleichung.

$$0 = \frac{1}{k} \ln s_{end} - s_{end} + 1$$

$$s_{end} = e^{k(s_{end}-1)} \tag{3.3}$$

Aus Gleichung (3.3) ist ersichtlich, dass s_{end} exponentiell mit k abnimmt. In der folgenden Tabelle ist der erwartete Anteil uninformierter Knoten nach Beendigung des Algorithmus in Abhängigkeit von k aufgelistet.

k	s_{end}
2	0.203
3	0.0595
4	0.0198
5	0.00698

Zu Beginn des Algorithmus steigt $i(s)$. Ab einem gewissen Punkt fällt die Funktion dann ab und geht gegen Null. Um das Maximum von $i(s)$ zu finden, wird die Gleichung 3.2 Null gesetzt. Daraus ergibt sich:

$$\frac{di}{dt} = 0 = kis - i$$
$$\frac{1}{k} = s$$

Dies bedeutet, dass die Anzahl der informierenden Knoten ihr Maximum erreicht hat, sobald $\frac{1}{k}$ der Knoten erreicht sind und in den folgenden Runden im Erwartungswert abnimmt.

3.3.2 Feedback Coin

Im Verlauf des Coin/Feedback Algorithmus versucht jeder infektiöse Knoten einen weiteren Knoten zu informieren. War dieser Zielknoten bereits zuvor informiert, so hört der Infektiöse mit Wahrscheinlichkeit $\frac{1}{k}$ auf, die Nachricht weiter zu verteilen (siehe Sektion 3.2).

Analog zur Analyse des Blind/Counter Algorithmus (Sektion 3.3.1) werden die Variablen s, i, r verwendet. Für den Verlauf von i und s können folgende Gleichungen aufgestellt werden.

$$\frac{ds}{dt} = -is \tag{3.4}$$
$$\frac{di}{dt} = is - \frac{1}{k}(1-s)i \tag{3.5}$$

Aus den Gleichungen (3.4) und (3.5) ergibt sich analog zu Sektion 3.3.1 die Funktion $i(s)$:

$$i(s) = \frac{k+1}{k}(1-s) + \frac{1}{k}\ln s$$

Daraus ergibt sich für s_{end}

$$s_{end} = e^{(k+1)(s_{end}-1)} \tag{3.6}$$

In der folgenden Tabelle ist der erwartete Anteil uninformierter Knoten nach Beendigung des Algorithmus in Abhängigkeit von k aufgelistet.

k	s_{end}
1	0.203
2	0.0595
3	0.0198
4	0.00698

Um das Maximum von $i(s)$ zu finden, wird die Gleichung (3.5) Null gesetzt. Daraus ergibt sich:

$$\begin{aligned} \frac{di}{dt} = 0 &= is - \frac{1}{k}(1-s)i \\ \frac{1}{k+1} &= s \end{aligned} \tag{3.7}$$

Dies bedeutet, dass sobald $\frac{1}{k+1}$ der Knoten erreicht sind die Anzahl der informierenden Knoten zurück geht.

3.3.3 Folgerung

Die Gleichungen (3.3) und (3.6) zeigen den erwarteten Anteil der Knoten, welche durch Blind/Counter, respektive Feedback/Coin erreicht werden. Es ist ersichtlich, dass mit einem Blind/Counter Verfahren, in dem jeder informierte Knoten k Nachrichten verschickt, im Erwartungswert ein gleichgrosser Teil des Netzes erreicht wird wie mit Feedback/Coin mit einer Abbruchwahrscheinlichkeit von $\frac{1}{k+1}$. Beide Erwartungswerte sind nicht abhängig von der Anzahl Knoten im Netz, sondern lediglich von dem Parameter k. Anhand des Blind/Counter Verfahrens soll nun berechnet werden, wie gross k sein muss, so dass im Erwartungswert das gesamte Netzwerk informiert wird.

Der nach Ablauf des Algorithmus uninformierte Anteil Knoten, s_{end} erfüllt Gleichung (3.3). Um alle Knoten zu erreichen muss s_{end} gegen 0 gehen. Setzt man in der Gleichung $s_{end} = 0$, so ergibt sich für k folgende Gleichung.

$$0 = e^{-k}$$

Da $\ln(0)$ nicht definiert ist, kann diese Gleichung nicht nach k aufgelöst werden. Dies kann so interpretiert werden, dass k unendlich gross sein muss, um im Erwartungswert alle Knoten zu erreichen.

Deshalb wird $s_{end} = n^{-x}$ gesetzt, für ein beliebig wählbares $x > 0$, so dass s_{end} für grosse n gegen Null strebt. Daraus ergibt sich folgende Gleichung.

$$\begin{aligned} n^{-x} &= e^{k(n^{-x}-1)} \\ \ln n^{-x} &= k(n^{-x}-1) \\ -x \ln x &= k(n^{-x}-1) \\ \frac{-x \ln n}{n^{-x}-1} &= k \end{aligned}$$

Für grosse n geht n^{-x} gegen 0 und folglich $n^{-x} - 1$ gegen -1. Daraus folgt:

$$\begin{aligned} k &= x \ln x \\ &= O(\ln n) \end{aligned}$$

Um im Erwartungswert alle Knoten mit einem Blind/Counter Algorithmus zu erreichen muss also $k = O(\ln n)$ gelten. Daraus folgt, dass k mindestens so gross sein muss um mit hoher Wahrscheinlichkeit alle Knoten zu erreichen.

Da sich das Verhalten von Feedback/Coin nur minimal von dem Verhalten von Blind/Counter unterscheidet, kann diese Erkenntnis auch auf Feedback/Coin übertragen werden.

3.4 Bekannte Resultate

In dieser Sektion werden zwei bekannte Gossiping Algorithmen vorgestellt. Beide gehen dabei von dem gleichen Netzwerkmodell aus. In diesem kennt jeder Teilnehmer des Netzwerks sämtliche anderen Knoten und kann direkt mit ihnen Kommunizieren. Die Topologie entspricht also einem kompletten Graphen.

3.4.1 Push Algorithmus

Wie der Name andeutet, verwendet dieser Algorithmus lediglich Kommunikation via Push. Der Push Algorithmus beginnt mit der Runde, in der ein Gerücht von einem Knoten generiert wird. In jeder darauf folgenden Runde schicken alle bereits informierten Knoten das Gerücht an einen zufällig ermittelten Kommunikationspartner weiter. Der Push Algorithmus geht von einem synchronen System aus und terminiert $O(\log n)$ Runden nachdem er begonnen hat. Alle infektiösen Knoten stellen dann das Versenden von Nachrichten ein. Insgesamt werden von dem Algorithmus $O(n \log n)$ Nachrichten verschickt.

Sei R die Anzahl der benötigten Runden, bis alle Teilnehmer des Netzwerks mit hoher Wahrscheinlichkeit über das Gerücht informiert sind. Boris Pittel [12] hat bewiesen, dass $R = \log_2 n + \log n + O(1)$.

3.4.2 Median-Counter Algorithmus

Karp et al. [9] stellen einen Gossiping Algorithmus vor, welcher insgesamt lediglich $O(n \log \log n)$ Nachrichten benötigt, um mit hoher Wahrscheinlichkeit alle Teilnehmer des Netzwerks zu erreichen. Wie der Push Algorithmus geht auch der Median-Counter Algorithmus von einem synchronen Modell aus und benötigt für die Verteilung der Information $O(\log n)$ Runden.

Für den Median-Counter Algorithmus wird das so genannte „Random Phone Call Model“ eingeführt. In diesem Modell nimmt in jeder Runde jeder am Netzwerk beteiligte Knoten Kontakt mit einem zufällig gewählten Kommunikationspartner auf. Dies geschieht unabhängig davon, ob der Knoten das Gerücht schon kennt oder nicht. Wenn nun ein Knoten das Gerücht kennt, teilt er es seinem Kommunikationspartner mit. Das Gerücht wird also sowohl via Push als auch via Pull verbreitet.

Im Verlauf des Algorithmus verändern die Knoten ihr Kommunikationsverhalten. Zu Beginn verteilt ein informierter Knoten das Gerücht mit Push und Pull. Später geht er dazu über, nur noch Pull-Kommunikation zu verwenden bis er schlussendlich die Verteilung ganz einstellt. Um zu wissen, wann er sein Verhalten ändern muss, führt jeder Knoten einen internen Zähler. In jeder Runde vergleicht ein Knoten seinen Zähler mit denen seiner Kommunikationspartner und zieht daraus Rückschlüsse über den Fortschritt der Verteilung. Gegebenenfalls gleicht er dann seinen eigenen Zähler nach einem bestimmten Verfahren an.

Im Unterschied zum Push Algorithmus wird beim Median-Counter Algorithmus das Gerücht nicht nur von den informierten Knoten aktiv verteilt, sondern jeder Knoten sucht gewissermassen nach Gerüchten, welche ihm noch unbekannt sind.

3.4.3 Asynchronität

Sowohl der Push als auch der Median-Counter Algorithmus gehen von einem synchronen Modell aus. In der Praxis ist so ein Modell aber nur sehr schwer oder gar nicht umsetzbar. Um praktikabel zu sein, sollte ein Gossiping Algorithmus in einem asynchronen Modell funktionieren.

Um den Push Algorithmus in einem asynchronen Modell anwenden zu können, kann ein *time-to-live* (ttl) Parameter eingeführt werden, welcher dem Gerücht angehängt wird. Am Anfang wird ttl auf $O(\log n)$ gesetzt. Jedes mal, wenn das Gerücht an einen Knoten geschickt wird, wird nun die ttl des Gerüchts dekrementiert. Sobald ttl null ist wird die Verteilung eingestellt.

Der Median-Counter Algorithmus kann nicht so leicht in einem asynchronen Modell angewandt werden. Für das Gelingen des Algorithmus ist es wichtig, dass auch uninformierte Knoten in jeder Runde einen zufälligen Kommunikationspartner ansprechen. Dies setzt eine Synchronisation der einzelnen Prozesse voraus. Auch sind die Vergleiche der Zähler abhängig von der jeweiligen Runde in der sie getätigt werden.

Kapitel 4

Gossiping in Peer-to-Peer

In diesem Kapitel wird eine kurze Einführung in Peer-to-Peer (P2P) Systeme gegeben. Es wird aufgezeigt welche Möglichkeiten für Gossiping in einem P2P-System bestehen und welche Probleme sich bei der Umsetzung eines Gossip-Verfahrens stellen.

4.1 Über Peer-to-Peer

Peer-to-Peer stellt eine Alternative zur traditionellen Client/Server Architektur dar. In einem Client/Server Modell macht der Client eine Anfrage an den Server, mit dem er verbunden ist. Der Server verarbeitet diese Anfrage und antwortet dem Client dann entsprechend.

In einem P2P Netzwerk wird ein einzelner Teilnehmer *Peer* - also *Ebenbürtiger* oder *Gleichgestellter* - genannt. Dieser Name deutet schon darauf hin, dass im Kontrast zum Client/Server Modell alle teilnehmenden Computer gleichberechtigt sind und sowohl als Client wie auch als Server fungieren können. Dies erlaubt es einem Peer sowohl neue Anfragen zu initiieren als auch auf Anfragen von anderen Peers im Netzwerk zu reagieren. Die Fähigkeit zum direkten Austausch von Information und Daten mit anderen Benutzern des Netzwerks, befreit P2P Benutzer von der Abhängigkeit von Servern. Dadurch wird ein P2P System vollkommen *dezentral*. Jeder Peer verfügt über die gleichen Rechte und Funktionalität.

Durch das Fehlen eines einzelnen Servers ist das System nahezu immun gegenüber von einzelnen Prozessen verursachte Flaschenhälse, da Daten und Last über das Netz verteilt werden. Auch kann durch den dezentralen Aufbau eines P2P Netzwerks der Ausfall einzelner Prozesse durchaus verkraftet werden, während in einem Client/Server System der Ausfall eines Servers gravierende Folgen haben kann.

Spätestens seitdem die MP3 Tauschbörse Napster um die Jahrtausendwende für Aufruhr gesorgt hat, ist P2P in aller Munde. Immerhin waren zu Spitzenzeiten bis zu 50 Millionen Benutzer durch Napster miteinander verbunden. Dies stellt wohl eine der grössten Urheberrechtsverletzungen der Geschichte dar.

Napster benötigte jedoch noch eine zentrale Datenbank. Der Benutzer musste zuerst mit einem Kontrollserver verbunden werden, auf welchem ein Datenverzeichnis mit allen verfügbaren Dateien angelegt war. Dort konnte der Benut-

zer nachschauen, welcher Peer die gewünschten Daten zur Verfügung stellt.

Modernere akademische P2P-Systeme, wie etwa Clippee [10], Chord [13] oder Kademlia [11], verwenden eine *Distributed Hash Table* (DHT), um Objekte anhand ihres Schlüssels in das P2P System einzufügen. Aufgrund dieses Schlüssels können die Daten dann im System auch wieder ausfindig gemacht werden.

Diese Diplomarbeit geht von einem solche P2P System aus und orientiert sich an einer Baumstruktur, wie sie beispielsweise Kademlia [11] verwendet. Es wird davon ausgegangen, dass jeder Peer mit $O(\log n)$ anderen Peers direkt verbunden ist (bei insgesamt n Peers). Diese sind seine Nachbarn.

4.2 Kommunikation in P2P

Jeder Knoten in einem P2P System ist nur mit $O(\log n)$ Nachbarn direkt verbunden. Mit diesen kann er direkt kommunizieren. Will ein Knoten A aber mit Knoten B kommunizieren, wobei B kein Nachbar von A ist, muss er die Nachricht erst an einen seiner Nachbarn schicken. Dieser leitet die Mitteilung dann über einen seiner Nachbarn weiter, bis die Nachricht Knoten B erreicht. Eine Mitteilung von einem Knoten zu einem anderen erzeugt also mehrere Nachrichten im Netzwerk.

In einem balancierten Kademlia Baum mit n Knoten geht der Weg zwischen zwei Knoten über maximal $\log n$ Verbindungen. Im Mittel ist ein Weg $\frac{1}{2}\log n$ Verbindungen lang. Um eine Information von einem Knoten zu einem anderen zu transferieren werden also $O(\log n)$ Nachrichten erzeugt.

In einem stark unbalancierten System kann ein Weg im schlimmsten Fall allerdings über bis zu $(n-1)$ Stationen gehen, da die Nachricht dann entlang einer linearen Liste der Knoten geschickt wird.

4.3 Umsetzung von Gossiping in P2P

Die in Sektion 3.4 vorgestellten Algorithmen gehen beide von einem kompletten Graphen als Netzwerkstruktur aus. In einer solche Topologie ist es nicht schwer, einen zufälligen Kommunikationspartner zu ermitteln: ein Knoten wählt aus allen ihm bekannten Knoten mit Wahrscheinlichkeit $\frac{1}{n}$ einen Partner aus. Da eine direkte Verbindung zwischen allen Knoten besteht, kann jeder Knoten problemlos mit diesem Partner kommunizieren.

In einem P2P-System kennt ein Knoten jedoch nur seine Nachbarn. Auch kann er lediglich mit seinen Nachbarn direkt und problemlos kommunizieren. Die restlichen Knoten des Netzwerks kennt er nicht und es kann auch nicht davon ausgegangen werden, dass er die Anzahl der Knoten kennt. Das Ermitteln eines zufällig gewählten Knotens des Netzwerks stellt also kein triviales Problem dar.

Um dennoch einen zufälligen Kommunikationspartner finden zu können, so dass für alle Knoten des Netzwerks die Wahrscheinlichkeit, gewählt zu werden, gleich gross ist, werden zwei mögliche Verfahren vorgestellt.

4.3.1 Zufälliger Knoten mittels Distributed Hash Table

In einer *Distributed Hash Table* (DHT) werden Daten, welche auf irgendeinem Knoten des Netzwerks gespeichert sind, mit Hilfe einer Hashfunktion lokalisiert.

Jeder am Netzwerk teilnehmende Knoten hat eine ID im Schlüsselraum der verwendeten Hashfunktion. Von jedem gespeicherten Datenobjekt wird ein Tupel `<Name, meine-ID>` erstellt, wobei *Name* der Name des Objekts und *meine-ID* die ID desjenigen Knotens ist, auf welchem das Objekt gespeichert ist. Dieses Tupel wird nun auf einem Knoten gespeichert, dessen ID sich in der „Nähe“ von `hash(Name)` befindet, für einen wohldefinierten Begriff der Nähe.

Sucht ein Benutzer nach dem Objekt, so berechnet er ebenfalls `hash(Name)` und schickt eine Anfrage an den Knoten, dessen ID in der Nähe des erhaltenen Wertes ist. Dieser kann dem suchenden Knoten dann mitteilen, wo das Objekt gespeichert ist. Wie dieses Verfahren im Detail aussehen kann, ist in [11] und [13] beschrieben.

In einem balancierten System ist jeder Knoten für die gleiche Anzahl Schlüssel zuständig. Wird nun in einem solchen balancierten System ein zufällig gewählter Schlüssel verwendet, um eine Nachricht an den für diesen Schlüssel zuständigen Knoten zu schicken, dann ist für jeden Knoten die Wahrscheinlichkeit, diese Nachricht zu erhalten, gleich gross. Vorausgesetzt der Schlüssel wurde mit uniformer Wahrscheinlichkeit gewählt.

Sobald das System aber unbalanciert ist und die Schlüssel ungleichmässig auf die Knoten verteilt sind, resultiert aus diesem Vorgehen keine uniforme Verteilung. Da in einem P2P System ständig neue Knoten hinzukommen und alte das Netzwerk verlassen kann nicht von einem balancierten System ausgegangen werden. Wie stark unbalanciert ein System sein kann, so dass dieses Vorgehen für einen Gossiping Algorithmus praktikabel ist, bleibt zu erforschen.

4.3.2 Zufälliger Knoten mittels DASIS

Der *Distributed Approximative System Information Service* (DASIS) [5] bietet approximative Information über ein P2P-System. Hierfür baut DASIS auf der Struktur des P2P-Systems auf. DASIS kann zum Beispiel verwendet werden, um die Anzahl der Knoten im Netzwerk zu ermitteln. Stark vereinfacht funktioniert DASIS wie folgt.

Jeder Knoten K teilt für sich das Netzwerk in Subdomänen auf. K ist der einzige Knoten des Netzwerks, welcher in keiner dieser Subdomänen ist. Alle anderen Knoten sind Mitglied von genau einer Subdomäne.

Jeder Nachbar von K ist dabei ein Experte für genau eine dieser Subdomänen. Er weiss zum Beispiel, wieviele Knoten in seiner Subdomäne enthalten sind. Dieses Wissen teilt er K mit. K weiss nun, wieviele Knoten in den Subdomänen seiner Nachbarn enthalten sind.

Um nun eine Nachricht an einen mit gleichmässiger Wahrscheinlichkeitsverteilung zufällig ermittelten Knoten zu schicken geht man wie folgt vor. Sei N_i ein Nachbar von K mit W_i Knoten in seiner Subdomäne. K schickt nun mit Wahrscheinlichkeit $p_i = \frac{W_i}{n}$ die Nachricht an den Nachbarn N_i. Die Wahrscheinlichkeit, dass die Nachricht an einen Nachbar geht, wird also mit der Anzahl Knoten der Subdomäne dieses Nachbars gewichtet. Derjenige, der die Nachricht bekommt, verschickt sie dann nach dem selben Prinzip an einen zufälligen Knoten in seiner Subdomäne weiter. Mit Wahrscheinlichkeit $\frac{1}{W_i}$ behält er sie jedoch selber.

DASIS liefert allerdings nur eine Approximation des Zustandes des Netzwerks. Die Wahrscheinlichkeit, der Empfänger der verschickten Nachricht zu sein, ist also nicht für alle Knoten genau gleich.

4.4 Push Algorithmus in Peer-to-Peer

In Sektion 4.3 wurde gezeigt, wie in einem P2P Netzwerk ein zufälliger Knoten mit uniformer Wahrscheinlichkeit gewählt werden kann. Somit kann der in Sektion 3.4.1 vorgestellte Push Algorithmus in einer P2P Umgebung umgesetzt werden.

Da in einem P2P Netzwerk die Kommunikationspartner aber nicht direkt miteinander kommunizieren, sondern die Nachrichten über ihre Nachbarn verschicken müssen, ist der Nachrichtenaufwand höher als in einem Netzwerk, in dem jeder Knoten mit jedem direkt verbunden ist.

In einem P2P Netzwerk geht eine Nachricht über $O(\log n)$ Knoten, bevor sie ihr Ziel erreicht. Daraus folgt, dass der Nachrichtenaufwand des Push Algorithmus in P2P $O(n \log^2 n)$ beträgt. In Kapitel 5 wird ein Gossiping Algorithmus für P2P vorgestellt, welcher mit $O(n \log n)$ Nachrichten auskommt.

Kapitel 5

Das Zwei-Phasen-Verfahren

In diesem Kapitel wird ein Gossiping Algorithmus für ein P2P Netzwerk vorgestellt. Es wird bewiesen, dass mit diesem Algorithmus mit hoher Wahrscheinlichkeit alle Knoten informiert werden.

5.1 Motivation der zwei Phasen

Eine Information muss zwischen zwei zufälligen Kommunikationspartnern über $O(\log n)$ Stationen geleitet werden. Wird der in Sektion 3.4.1 vorgestellte Algorithmus, wie in Sektion 4.4 gezeigt, für ein P2P Netzwerk adaptiert, so entsteht durch die langen Kommunikationswege ein erheblicher Mehraufwand an Nachrichten. Insgesamt werden $O(n \log^2 n)$ Nachrichten versandt.

Die Idee des in diesem Kapitel vorgestellten Verfahrens ist es, zunächst einen Teil der Knoten mit dem Gossiping Verfahren zu informieren. Danach verteilen die in dieser Weise informierten Knoten die Information an solche Knoten weiter, die mit wenigen Stationen erreicht werden können.

5.2 Das Verfahren

Bevor das Zwei-Phasen-Verfahren vorgestellt wird, wird zunächst die Bedeutung von *globaler Verteilung* und *lokaler Verteilung* im Kontext dieser Arbeit festgelegt.

globale Verteilung: Mit diesem Ausdruck wird das Verteilen der Information an einen beliebigen Knoten des Netzwerks bezeichnet. Die Entfernung der beiden Knoten im Netzwerk ist hierbei abhängig von der Topologie des Netzwerks.

lokale Verteilung: Hiermit wird das Verteilen der Information an Knoten in der unmittelbaren Nähe des Versenders bezeichnet. Der Adressat kann also vom Sender über nur wenige Knoten erreicht werden.

Das Zwei-Phasen-Verfahren wendet nun globale und lokale Verteilung an, um eine Information mit hoher Wahrscheinlichkeit jedem Knoten des Netzwerks mitzuteilen. Hierzu wird die Information in einer ersten Phase global an einen festen Anteil der Knoten verteilt. Diese Verteilung geschieht mit einem der in

Sektion 4.3 vorgestellten Verfahren. Für jeden Knoten ist die Wahrscheinlichkeit, Empfänger einer versandten Nachricht zu sein, gleich gross. Nach Abschluss der ersten Phase ist demnach für jeden Knoten die Wahrscheinlichkeit, informiert zu sein, gleich gross. Die Information ist also gleichmässig über das Netzwerk verteilt.

In der zweiten Phase wird die Information von denjenigen Knoten, welche in der ersten Phase informiert wurden, lokal verbreitet. Dies geschieht, indem die informierten Knoten die Information an alle Knoten schicken, die sie mittels zweier Verbindungen erreichen können. Hierzu wird die Information direkt an alle Nachbarn geschickt. Diese wiederum informieren ebenfalls ihre Nachbarn.

Ein Knoten bleibt nach Ausführung beider Phasen uninformiert, wenn in der ersten Phase weder er selber, noch seine lokale Umgebung informiert wurden. Im Folgenden wird bewiesen, dass mit hoher Wahrscheinlichkeit alle Knoten nach Ablauf beider Phasen informiert sind.

5.3 Erste Phase: Globale Verteilung

Für das Gelingen des Zwei Phasen Algorithmus ist es wichtig, dass mit der ersten Phase ein fester Anteil c der Knoten mit hoher Wahrscheinlichkeit erreicht wird. Um die Verteilung der Information besser untersuchen zu können, wird ein vereinfachtes Modell der Verteilung eingeführt. Der Verteilungsvorgang wird in Runden unterteilt. Ein infektiöser Knoten schickt alle seine Nachrichten in einer Runde, danach hört er auf zu senden und wird inaktiv. Die Knoten, welche in einer Runde t neu informiert werden, sind dann in der Runde $t + 1$ infektiös.

Im Folgenden wird für einen Blind-Counter Algorithmus mit $k = 2$ gezeigt, dass mit hoher Wahrscheinlichkeit die Anzahl infektiöser Knoten I in jeder Runde um einen Faktor $d = 1.2$ steigt, bis ein Anteil von $c = \frac{1}{10}$ der Knoten erreicht ist. Dadurch wird implizit gezeigt, dass mit hoher Wahrscheinlichkeit $\frac{1}{10}$ der Knoten erreicht werden.

Für den Beweis wird der Verlauf der ersten Phase in zwei Stadien unterteilt. Im Startstadium werden die ersten vier Runden betrachtet. Für die weitere Verteilung wird die Wahrscheinlichkeit für die darauf folgenden Runden abgeschätzt und dann aufsummiert.

5.3.1 Startstadium

Es soll gezeigt werden, dass die Anzahl infektiöser Knoten in jeder Runde um einen festen Faktor $d = 1.2$ wächst. Allerdings können Knoten nicht zu einem Bruchteil informiert werden. Es muss also davon ausgegangen werden, dass die Anzahl in einer Runde neu informierter Knoten abgerundet wird. In den ersten Runden sind nur wenige Knoten infektiös. Wird hier die Anzahl neu informierter Knoten abgerundet, findet keinerlei Wachstum statt. Aus diesem Grund wird für die ersten Runden gezeigt, dass mit hoher Wahrscheinlichkeit ein echtes Wachstum um mindestens Faktor $d = 1.2$ stattfindet. Im Folgenden bezeichnet $P_t(x)$ die Wahrscheinlichkeit, dass in Runde t genau x neue Knoten informiert werden. Dies unter der Bedingung, dass in Runde $t - 1$ die Anzahl infektiöser Knoten um $d = 1.2$ gewachsen ist.

Das Startstadium beinhaltet die ersten vier Runden der Verteilung. In der ersten Runde ist genau ein Knoten infektiös. Das ist der Knoten, welcher des Gerücht in Umlauf bringen will. Damit die Anzahl der infektiösen Knoten um den Faktor $d = 1.2$ steigt, müssen in dieser Runde zwei Knoten neu informiert werden. Da genau zwei Nachrichten verschickt werden, müssen beide einen uninformierten Knoten erreichen.

1. Runde

Die Wahrscheinlichkeit $P_1(2)$, dass in der ersten Runde zwei neue Knoten informiert werden beträgt:

$$P_1(d \geq 1.2) = P_1(2) = \frac{n-1}{n}\frac{n-2}{n} = 1 - \frac{3n-2}{n^2} = 1 - O(n^{-1}) \tag{5.1}$$

Danach sind zwei Knoten infektiös und ein Knoten inaktiv. In der zweiten Runde werden also vier Nachrichten verschickt.

2.Runde

Damit die Anzahl der infektiösen Knoten mindestens um den Faktor 1.2 steigt, müssen mindestens drei der insgesamt vier versandten Nachrichten an bisher uninformierte Knoten gehen.

Die Wahrscheinlichkeit $P_2(3)$, dass in der zweiten Runde drei neue Knoten informiert werden, beträgt:

$$P_2(3) = \frac{(n-3)(n-4)(n-5)}{n^3}\frac{(3+4+5+6)}{n}$$

Die Wahrscheinlichkeit $P_2(4)$, dass in der zweiten Runde vier neue Knoten erreicht werden, ist:

$$P_2(4) = \frac{(n-3)(n-4)(n-5)(n-6)}{n^4}$$

Aus den Wahrscheinlichkeiten $P_2(3)$ und $P_2(4)$ ergibt sich dann für die Wahrscheinlichkeit, dass in der zweiten Runde die Anzahl infektiöser Knoten mindestens mit dem Faktor 1.2 steigt, $P_3(d \geq 1.2)$:

$$P_2(d \geq 1.2) = P_2(3) \cup P_2(4) = P_2(3) + P_2(4) = 1 - \frac{97n^2 - 504n + 720}{n^4}$$

$$P_2(d \geq 1.2) = 1 - O(n^{-2}) \tag{5.2}$$

In der zweiten Runde werden also mit hoher Wahrscheinlichkeit mindestens drei neue Knoten informiert. Die Wahrscheinlichkeit eines Wachstumsfaktors von 1.2 in der dritten Runde wird nun unter der Bedingung abgeschätzt, dass drei Knoten infektiös sind.

3. Runde

In der dritten Runde sind also mindestens drei Knoten infektiös und drei Knoten informiert und inaktiv. Demzufolge werden sechs Nachrichten verschickt. Damit ein Wachstum der infektiösen Knoten um mindestens Faktor 1.2 stattfindet, müssen mindestens fünf Knoten neu informiert werden. Für die Abschätzung der Wahrscheinlichkeit wird die Gegenwahrscheinlichkeit berechnet. Dazu werden die Wahrscheinlichkeiten null, einen, zwei, drei oder vier Knoten zu erreichen abgeschätzt. Dies geschieht mittels einer binomischen Verteilung.

Sei p_t die Wahrscheinlichkeit, dass eine Nachricht einen neuen Knoten informiert und $\bar{p}_t = 1 - p_t$. Dann gilt für die Wahrscheinlichkeit $B_M(x)$, dass von insgesamt M verschickten Nachrichten genau x einen neuen Knoten informieren folgende Gleichung.

$$B_M(x) = \binom{M}{x} p_t^x * \bar{p}_t^{M-x}$$

Damit die Wahrscheinlichkeiten keinen, einen, zwei, drei oder vier Knoten zu erreichen zu gross abgeschätzt werden, muss p_t zu gross gewählt werden. Da zu Beginn der dritten Runde sechs Knoten informiert sind, wird $p_t = \frac{n-6}{n}$ gesetzt. Dementsprechend wird $\bar{p}_t$ zu klein. Dadurch wird die Wahrscheinlichkeit, zu wenige Knoten neu zu informieren, zu gross und die Wahrscheinlichkeit, genug Knoten zu erreichen, zu klein abgeschätzt.

Die Wahrscheinlichkeiten $P_3(x)$, dass in der dritten Runde nur x neue Knoten informiert werden, für alle $x < 5$ ist

$$\begin{aligned}
P_3(0) &\leq \left(\frac{6}{n}\right)^6 = O(n^{-6}) \\
P_3(1) &\leq \binom{6}{1}\left(\frac{n-6}{n}\right)\left(\frac{6}{n}\right)^5 = O(n^{-5}) \\
P_3(2) &\leq \binom{6}{2}\left(\frac{n-6}{n}\right)^2\left(\frac{6}{n}\right)^4 = O(n^{-4}) \\
P_3(3) &\leq \binom{6}{3}\left(\frac{n-6}{n}\right)^3\left(\frac{6}{n}\right)^3 = O(n^{-3}) \\
P_3(4) &\leq \binom{6}{4}\left(\frac{n-6}{n}\right)^4\left(\frac{6}{n}\right)^2 = O(n^{-2})
\end{aligned}$$

Daraus ergibt sich die Wahrscheinlichkeit, dass die Anzahl informierter Knoten in der dritten Runde mindestens um den Faktor 1.2 steigt:

$$\begin{aligned}
P_3(d \geq 1.2) &= 1 - \sum_{x=0}^{4} P_3(x) \\
P_3(d \geq 1.2) &= 1 - O(n^{-2}) \qquad (5.3)
\end{aligned}$$

4. Runde

In der vierten Runde sind fünf Knoten infektiös und sechs Knoten inaktiv. Es werden zehn Nachrichten verschickt, von denen mindestens sechs treffen müssen

um einen Wachstum der infektiösen Knoten um Faktor 1.2 zu erreichen. Die Abschätzung der vierten Runde geschieht analog zu der dritten Runde. Nun gilt aber $p_t = \frac{n-11}{n}$ und demzufolge $\bar{p}_t = \frac{11}{n}$. Daraus ergeben sich folgende Abschätzungen.

$$
\begin{aligned}
P_4(0) &\leq \left(\frac{11}{n}\right)^{10} = O(n^{-10}) \\
P_4(1) &\leq \binom{10}{1}\left(\frac{n-11}{n}\right)\left(\frac{11}{n}\right)^{9} = O(n^{-9}) \\
P_4(2) &\leq \binom{10}{1}\left(\frac{n-11}{n}\right)^{2}\left(\frac{11}{n}\right)^{8} = O(n^{-8}) \\
P_4(3) &\leq \binom{10}{1}\left(\frac{n-11}{n}\right)^{3}\left(\frac{11}{n}\right)^{7} = O(n^{-7}) \\
P_4(4) &\leq \binom{10}{1}\left(\frac{n-11}{n}\right)^{4}\left(\frac{11}{n}\right)^{6} = O(n^{-6}) \\
P_4(5) &\leq \binom{10}{1}\left(\frac{n-11}{n}\right)^{5}\left(\frac{11}{n}\right)^{5} = O(n^{-5})
\end{aligned}
$$

Daraus ergibt sich die Wahrscheinlichkeit, dass die Anzahl informierter Knoten in der vierten Runde um den Faktor 1.2 steigt:

$$
\begin{aligned}
P_4(d \geq 1.2) &= 1 - \sum_{j=5}^{10} O(n^{-j}) \\
P_4(d \geq 1.2) &= 1 - O(n^{-5}) \qquad (5.4)
\end{aligned}
$$

Wahrscheinlichkeit über die ersten vier Runden

Um die Wahrscheinlichkeit $P_{Start}(d \geq 1.2)$, dass die Anzahl infektiöser Knoten während des gesamten Startstadiums in jeder Runde mindestens um den Faktor 1.2 steigt, zu berechnen, wird der Multiplikationssatz der Wahrscheinlichkeitsrechnung (5.5) angewandt.

$$P(A \cap B) = P(A) * P(B|A) \qquad (5.5)$$

Da die Wahrscheinlichkeit für die Runde t unter der Bedingung berechnet wurde, dass in Runde $t-1$ das Wachstum von I genügend gross war, ist nach (5.5) $P_{Start}(d \geq 1.2)$ gleich dem Produkt der Wahrscheinlichkeiten der ersten vier Runden.

$$P_{Start}(d \geq 1.2) = \prod_{t=1}^{4} P_t(d \geq 1.2) = 1 - O(n^{-1}) \qquad (5.6)$$

$1 - O(n^{-1})$ entspricht einer hohen Wahrscheinlichkeit. Es werden in den ersten vier Runden also mit hoher Wahrscheinlichkeit 17 Knoten informiert. In der fünften Runde sind dann sechs der Knoten infektiös und verschicken insgesamt zwölf Nachrichten. Elf Knoten sind bereits informiert und nun inaktiv.

5.3.2 Weitere Verteilung

In diesem Abschnitt wird gezeigt, dass mit hoher Wahrscheinlichkeit die Anzahl infektiöser Knoten in jeder Runde mit einem Faktor $d = 1.2$ steigt. Dies so lange, bis insgesamt ein Anteil $c = \frac{1}{10}$ aller Knoten erreicht ist.

Um dies zu zeigen wird die Wahrscheinlichkeit $P_t(d < 1.2)$, dass die Anzahl infektiöser Knoten in der Runde t nicht um den Faktor d steigt, zu jeder Runde abgeschätzt. Es wird gezeigt, dass $P_t(d < 1.2)$ sehr klein ist, solange der Anteil c der Knoten noch nicht erreicht ist. Schliesslich werden dann alle Wahrscheinlichkeiten, dass kein genügendes Wachstum stattfindet, aufsummiert. Daraus ergibt sich dann die gesuchte Wahrscheinlichkeit, dass in allen Runden ein hinreichendes Wachstum stattfindet.

Die Wahrscheinlichkeit der einzelnen Runden wird mittels einer Chernoff Schranke [8] abgeschätzt. Eine Chernoff Schranke begrenzt die Wahrscheinlichkeit, dass sich eine Zufallsvariable Y im „Schwanz“ einer Verteilung befindet - also weit entfernt von Erwartungswert. Die Chernoff Schranke kann für alle Poisson Verteilungen angewandt werden. Es gibt Chernoff Schranken sowohl für den oberen als auch für den unteren Schwanz einer Verteilung. Für diese Rechnung wird jedoch nur die obere Schranke (5.7) verwendet .

$$Pr[Y > (1+\delta)\mu] < \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^{\mu} \tag{5.7}$$

Um die Chernoff Schranke für jede Runde berechnen zu können, muss also der Erwartungswert μ und das δ, welches angibt wie weit der Schwanz vom Erwartungswert entfernt ist, berechnet werden.

Zunächst wird der Erwartungswert μ berechnet. Hierfür muss bekannt sein wieviele Nachrichten in einer Runde verschickt werden und wie gross die Wahrscheinlichkeit für eine Nachricht ist, dass sie an einen bereits informierten Knoten gesandt wird.

Die Anzahl der infektiösen Knoten in einer Runde kann einfach über eine geometrische Reihe berechnet werden. Da von einem fixen Wachstum $d = 1.2$ ausgegangen wird und in der ersten Runde genau ein Knoten infektiös ist, gilt für die Anzahl infektiöser Knoten I in der Runde t: $I(t) = d^{(t-1)}$. Es gilt also für die Anzahl der in Runde t abgeschickten Nachrichten $M(t)$:

$$M(t) = 2d^{(t-1)}. \tag{5.8}$$

Gezeigt werden soll, dass nicht mehr als ein bestimmter Grenzwert der Nachrichten an bereits informierte Knoten geht. Deshalb muss die Abschätzung der Wahrscheinlichkeit $\bar{p}_t$, dass eine Nachricht einen zuvor informierten Knoten erreicht, zu gross gewählt werden. Sei $R(t)$ die Anzahl der informierten Knoten, welche die Verteilung eingestellt haben und sei $\bar{S}(t) = R(t)+I(t)$ die Anzahl der in Runde t informierten Knoten. Da dies der Summe aller infektiösen Knoten der vorhergehenden Runden entspricht, ergibt sich aus der Summenformel für geometrische Reihen $\bar{S}(t) = \frac{1-d^t}{1-d}$.

Wie in Sektion 3.3.1 bezeichnen s, i und r die Bruchteile der Knoten die uninformiert, infektiös bzw informiert und inaktiv sind. Die Wahrscheinlichkeit, dass eine Nachricht in Runde t einen bereits informierten Knoten erreicht, ist also gleich $1 - s(t) = \bar{s}(t)$ (Abbildung 5.3.2a). Allerdings kann es auch sein, dass zwei oder mehr Nachrichten an den gleichen bis anhin uninformierten Knoten

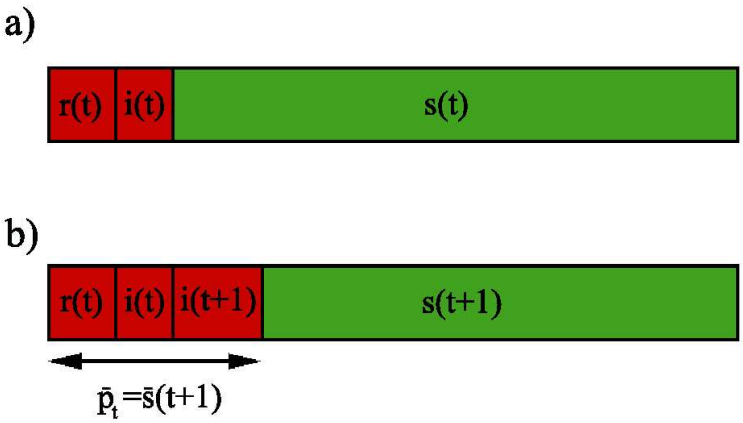

Abbildung 5.1: Abschätzung der Wahrscheinlichkeit, dass eine Nachricht in Runde t keinen neuen Knoten informiert.

gehen. Also gilt $\bar{p}_t > \bar{s}(t)$. Aus diesem Grund werden diejenigen Nachrichten, die an Knoten gehen, die in dieser Runde neu informiert werden, ebenfalls zu den Nachrichten gezählt, die keinen neuen Knoten informieren (Abbildung 5.3.2b). Es gilt also für $\bar{p}_t$:

$$\begin{aligned} \bar{p}_t &\leq r(t) + i(t) + i(t+1) = r(t+1) + i(t+1) = \bar{s}(t+1) \\ \bar{p}_t &\leq \frac{1}{n}\bar{S}(t+1) = \frac{1}{n}\frac{1-d^{(t+1)}}{1-d} \end{aligned} \tag{5.9}$$

Aus (5.9) lässt sich der Erwartungswert $\mu(t)$, wieviele Nachrichten in Runde keine neuen Knoten informieren, berechnen.

$$\mu(t) = M(t) * \bar{p}_t = \frac{10}{n} d^{(t-1)}(d^{(t+1)} - 1) \tag{5.10}$$

In Runde t werden $2I(t)$ Nachrichten verschickt. Um das geforderte Wachstum um Faktor 1.2 zu garantieren müssen mindestens $1.2I(t)$ Nachrichten neue Knoten informieren. Es dürfen also nicht mehr als $0.8I(t)$ Nachrichten an bereits informierte Knoten geschickt werden. Daraus ergeben sich folgende Gleichungen für $\delta(t)$.

$$(1 + \delta(t))\mu(t) = 0.8d^{(t-1)}$$

$$\delta(t) = 0.08n\frac{1}{(d^{(t+1)} - 1)} - 1 \tag{5.11}$$

Die Werte aus (5.10) und (5.11) in die Chernoff Gleichung (5.7) eingesetzt ergibt die gesuchte Wahrscheinlichkeit $P_c(t)$.

$$P_c(t) \leq \left(\frac{e^{\frac{0.08n}{(d^{(t+1)}-1)}-1}}{\left(\frac{0.08n}{d^{(t+1)}-1}\right)^{\left(\frac{0.08n}{d^{(t+1)}-1}\right)}} \right)^{\frac{10}{n}d^{(t-1)}(d^{(t+1)}-1)} \tag{5.12}$$

Umgeformt ergibt dies einen Ausdruck mit zwei Faktoren.

$$P_c(t) \leq \left(\frac{1}{e}\right)^{\frac{10}{n}d^{(t-1)}(d^{(t+1)}-1)} \left(\frac{e(d^{(t+1)}-1)}{0.08n}\right)^{0.8d^{(t-1)}} \tag{5.13}$$

$$\tag{5.14}$$

Seien $P_l(t)$ und $P_r(t)$ wie folgt definiert.

$$P_l(t) := \left(\frac{1}{e}\right)^{\frac{10}{n}d^{(t-1)}(d^{(t+1)}-1)}$$

$$P_r(t) := \left(\frac{e(d^{(t+1)}-1)}{0.08n}\right)^{0.8d^{(t-1)}}$$

Es gilt dann für $P_c(t)$:

$$P_c(t) \leq P_l(t) * P_r(t)$$

Zu zeigen ist nun, dass $P_c(t)$ für grosse n sehr klein wird, solange noch nicht $\frac{1}{10}$ der Knoten erreicht ist. In Sektion (5.3.1) wurde gezeigt, dass bis $t = 4$ mit hoher Wahrscheinlichkeit ein Wachstum der infektiösen Knoten um Faktor 1.2 stattfindet. Der linke Faktor $P_l(t)$ der Gleichung (5.13) wird für alle $t > 4$ kleiner als 1. Da $P_c(t)$ also durch $P_l(t)$ kleiner kleiner wird, reicht es zu zeigen, dass der rechte Faktor $P_r(t)$ für grosse n sehr klein wird. Um dies zu zeigen wird das Maximum dieses Terms im betrachteten Intervall gesucht. Zu diesem Zweck wird die zweite Ableitung betrachtet.

Da $I(t) = d^{(t-1)}$ kann $P_r(t)$ auch wie folgt dargestellt werden.

$$P_r(t) = \left(\frac{e(1.44I(t)-1)}{0.08n}\right)^{0.8I(t)} \tag{5.15}$$

Für die zweite Ableitung von (5.15) nach $I(t)$ ergibt das:

$$\begin{aligned}
\frac{dP_r(I(t))}{d^2I(t)} \;=\; & \left(12.5\,\frac{e\,(1.44\,I(t)-1)}{n}\right)^{0.8\,I(t)} * \\
& * \left(0.8\,\ln\left(12.5\,\frac{e\,(1.44\,I(t)-1)}{n}\right) + 1.152\,\frac{I(t)}{1.44\,I(t)-1}\right)^2 + \\
& + \left(12.5\,\frac{e\,(1.44\,I(t)-1)}{n}\right)^{0.8\,I(t)} * \\
& * \left(2.304\,(1.44\,I(t)-1)^{-1} - 1.65888\,\frac{I(t)}{(1.44\,I(t)-1)^2}\right)
\end{aligned} \tag{5.16}$$

Da in Sektion (5.3.1) das Verhalten für $I(t) \leq 6$ bereits untersucht wurde, reicht es das Verhalten für $I(t) \geq 6$ zu betrachten. Es gilt:

$$\forall I(t) \geq 6 : \quad \left(2.304 \left(1.44\, I(t) - 1\right)^{-1} - 1.65888 \frac{I(t)}{\left(1.44\, I(t) - 1\right)^2} \right) > 0 \quad (5.17)$$

(5.17) ist der einzige Term von (5.16), der negativ werden kann. Da er für alle $I(t) \geq 6$ positiv ist, folgt, dass die zweite Ableitung für alle $I(t) \geq 6$ positiv ist.

Es existiert demnach kein lokales Maximum. Das Maximum von $P_r(t)$ ist also entweder bei $I(t) = 6$ oder bei $I(t_c)$, wobei t_c die Runde bezeichnet, in der $c = \frac{1}{10}$ der Knoten erreicht werden.

Setzt man $I(t) = 6$ ergibt sich gemäss 5.15 für $P_r(6)$:

$$P_r(6) = 3.19 * 10^9 \left(\frac{e}{n}\right)^{4.8} = O\left(\frac{1}{n^{4.8}}\right) \quad (5.18)$$

Um $P_r(t_c)$ berechnen zu können, muss zuerst $I(t_c)$ berechnet werden. Aus $\bar{S}(t) = \frac{1-d^t}{1-d}$ und $I(t) = d^{(t-1)}$ ergibt sich für $I(t_c)$:

$$\begin{aligned} \frac{1}{10}n &= \frac{1-d^{t_c}}{1-d} \\ 0.02n + 1 &= d^{t_c} \\ d^{t_c} &= 1.2 * I(t_c) \\ I(t_c) &= \frac{1}{60}n + \frac{5}{6} \end{aligned} \quad (5.19)$$

Aus 5.15 und 5.19 ergibt sich für $P_r(t_c)$:

$$P_r(t_c) = \left(\frac{12.5e(0.024n + 0.2)}{n}\right)^{(\frac{1}{70}n + \frac{2}{3})} = O(0.8^{(\frac{1}{70}n)}) \quad (5.20)$$

Für grosse n gilt $O(n^{-4.8}) > O(0.8^{(\frac{1}{70}n)})$. Die Wahrscheinlichkeit, dass die Anzahl infektiöser Knoten nicht um den Faktor 1.2 steigt, ist also bei $I(t) = 6$ am grössten. Es kann also angenommen werden, dass die Wahrscheinlichkeit zu jedem Zeitpunkt der Verteilung höchstens so gross ist wie in Gleichung (5.18) angegeben, bis $\frac{1}{10}$ der Knoten erreicht sind. Es gilt also für $P_t(d < 1.2)$:

$$P_t(d < 1.2) = O\left(\frac{1}{n^{4.8}}\right) \quad (5.21)$$

5.3.3 Laufzeit

Gleichung (5.18) ist die Abschätzung, dass die Anzahl infektiöser Knoten in einer Runde um den geforderten Faktor steigt. Um die Gesamtwahrscheinlichkeit über alle Runden zu erhalten, muss zunächst die benötigte Anzahl Runden berechnet werden.

Um zu berechnen, wieviele Runden R benötigt werden, bis ein Anteil c der Knoten erreicht ist, kann mit Hilfe des Wachstumsfaktors $d = 1.2$ eine geometrische Reihe aufgestellt werden.

$$1 + d + d^2 + \ldots + d^R = cn = \frac{1 - d^R}{1 - d}$$

Diese Reihe kann nun nach R aufgelöst werden.

$$\begin{aligned} d^R &= 1 - cn(1 - d) \\ R &= log_d(1 - cn(1 - d)) \end{aligned} \tag{5.22}$$

Setzt man $d = 1.2$ und $c = \frac{1}{10}$ in Gleichung (5.22) ein, Erhält man für die Anzahl benötigter Runden R:

$$R = log_{1.2}\left(1 + \frac{1}{5}n\right) = O(\log n) \tag{5.23}$$

Dies ist die benötigte Anzahl Runden, wenn I in jeder Runde genau um den Faktor $d = 1.2$ ansteigt. Allerdings kann in einem realen Vorgang I nur natürliche Werte annehmen. Ein Knoten kann nicht nur zu einem Bruchteil informiert sein. Dementsprechend muss bewiesen werden, dass, wenn die Anzahl in einer Runde t neu informierter Knoten abgerundet wird, immernoch lediglich $O(\log n)$ Runden insgesamt benötigt werden.

Um dies zu zeigen genügt es folgende Gleichung zu beweisen.

$$\lfloor 1.2 * \lfloor 1.2I \rfloor \rfloor \geq 1.2I \tag{5.24}$$

Aus (5.24) folgt, dass höchstens die doppelte Anzahl Runden benötigt wird, also immernoch $O(\log n)$. Nun soll (5.24) bewiesen werden.

Bei der Operation $\lfloor 1.2I \rfloor$ geht jeweils ein Wert λ verloren. Für λ gilt dabei:

$$0 \leq \lambda \leq 1$$

Um (5.24) zu zeigen genügt es also zu zeigen, dass $\lfloor 1.2I \rfloor \geq I + 1$. Sei δ wie folgt definiert.

$$\delta(I) = \lfloor 1.2I \rfloor - I$$

Zu zeigen ist also $\delta(I) \geq 1$, für alle $I \geq 6$. Für $6 \leq I \leq 10$ ergibt $\delta(I)$ folgende Werte.

$$\begin{aligned} \delta(6) &= \lfloor 7.2 \rfloor - 6 = 1 \\ \delta(7) &= \lfloor 8.4 \rfloor - 7 = 1 \\ \delta(8) &= \lfloor 9.6 \rfloor - 8 = 1 \\ \delta(9) &= \lfloor 10.8 \rfloor - 9 = 1 \\ \delta(10) &= \lfloor 12 \rfloor - 10 = 2 \end{aligned}$$

Da $\delta(I)$ eine monoton wachsende Funktion ist folgt daraus, dass in jeweils zwei Runden immer ein reales Wachstum um mindestens den Faktor 1.2 stattfindet. Insgesamt werden also $O(\log_{1.2} n)$ Runden benötigt, um $\frac{1}{10}$ der Knoten zu erreichen.

Zusammen mit (5.18) ergibt sich daraus für die Wahrscheinlichkeit, dass in allen dieser Runden genügend neue Knoten informiert werden, folgender Ausdruck.

$$P_{weitere} = 1 - O\left(\frac{\log_{1.2} n}{n^{4.8}}\right) \tag{5.25}$$

5.3.4 Gesamte Wahrscheinlichkeit der ersten Phase

Die Wahrscheinlichkeiten, dass während des Startstadiums genügend Knoten erreicht werden (5.6) und während der weiteren Verteilung die Zunahme der infektiösen Knoten genügend gross ist (5.25), sind nun bekannt. Aus ihnen folgt die Wahrscheinlichkeit $P_{0.1}$, dass mindestens $\frac{1}{10}$ der Knoten erreicht werden.

$$P_{0.1} \geq \left(1 - O\left(n^{-1}\right)\right)\left(1 - O\left(\frac{\log_{1.2} n}{n^{4.8}}\right)\right) = 1 - O\left(n^{-1}\right) \tag{5.26}$$

$P_{0.1}$ entspricht einer „hohen Wahrscheinlichkeit“. Des weiteren ist $P_{0.1}$ gleich der Wahrscheinlichkeit, dass in der ersten Runde genügend Knoten informiert werden (5.1). Um also die Gesamtwahrscheinlichkeit zu erhöhen genügt es, wenn der erste Knoten mehr als zwei Nachrichten verschickt. Dadurch steigt die Wahrscheinlichkeit, dass in der ersten Runde mindestens zwei Knoten neu informiert werden.

5.4 Zweite Phase: Lokale Verteilung

In Sektion (5.3) wurde gezeigt, dass mit einem einfachen Push Algorithmus mit hoher Wahrscheinlichkeit mindestens $\frac{1}{10}$ der Knoten informiert werden. Da die Information an unabhängig und gleichmässig zufällige Knoten geschickt wurde, kann man davon ausgehen, dass nach der globalen Verteilung für jeden Knoten N gilt:

$$\forall N: \quad \Pr[\text{N ist nicht informiert}] \leq \frac{9}{10}$$

In einem typischen Peer-2-Peer Netzwerk hat ein Knoten $O(\log n)$ Nachbarn. Dementsprechend kann ein Knoten $O(\log^2 n)$ andere Knoten mit nur zwei Hops erreichen. Umgekehrt kann ein Knoten aber auch von $O(\log^2 n)$ anderen Knoten mit nur zwei Hops erreicht werden. Wenn nun in der lokalen Verteilung ein in Phase 1 informierter Knoten die Nachricht an alle Knoten in Entfernung zwei weiter schickt, ist die Wahrscheinlichkeit, dass ein Knoten N nicht informiert wird gleich der Wahrscheinlichkeit, dass kein Knoten in Entfernung zwei informiert ist.

$$Pr[\text{N wird informiert}] = 1 - O\left(\left(\frac{9}{10}\right)^{\log^2 n}\right) \tag{5.27}$$

Generell gilt für alle k, p, so dass $0 < p < 1$ und $k \geq 1$:

$$1 - p \leq \left(1 - \frac{p}{k}\right)^k \tag{5.28}$$

Aus (5.27) und (5.28) folgt für die Wahrscheinlichkeit, dass alle Knoten informiert werden:

$$Pr[\text{Alle Knoten informiert}] \geq 1 - O\left(n\left(\frac{9}{10}\right)^{\log^2 n}\right) \tag{5.29}$$

Für grosse n entspricht (5.29) einer hohen Wahrscheinlichkeit. Es werden also mit hoher Wahrscheinlichkeit alle Knoten des Netzwerks mit dem beschrieben Vorgehen erreicht.

5.5 Diskussion

Wie Gleichung (5.29) zeigt, sind nach Ablauf des Algorithmus mit hoher Wahrscheinlichkeit alle Knoten informiert. Insgesamt werden im Laufe der Verteilung $O(n \log n)$ Nachrichten verschickt. In einer synchronen Umgebung benötigt der Algorithmus hierfür $O(\log^2 n)$ Runden.

Es ist nicht notwendig, dass die zwei Phasen der Verteilung zeitlich voneinander getrennt sind. Ein Teil der Knoten kann noch an der globalen Verteilung arbeiten, während der Rest mit der lokalen begonnen hat. Ein Knoten muss lediglich wissen, in welcher Phase er die Information bekommen hat. Erhält er die Nachricht in der ersten Phase, so muss er sie auch global weiter verteilen. Erhält er sie in der zweiten Phase, so ist es nicht notwendig, dass er die Information noch global verteilt.

Das Zwei-Phasen-Verfahren ist nicht von einer synchronen Umgebung abhängig, sondern funktioniert auch im asynchronen Fall. Jeder Knoten agiert unabhängig von den restlichen Knoten. Wird ein Knoten informiert, schickt er gemäss dem Algorithmus seine Nachrichten ab.

Kapitel 6

Vergleich von Broadcast Algorithmen in P2P

In einem Broadcast-Szenario gibt es einen Ausgangsknoten, welcher eine Nachricht oder eine Information an alle anderen Knoten im selben Netzwerk verschicken will. In diesem Kapitel werden verschiedene Algorithmen zur Lösung des Broadcast-Problems in einem typischen Peer-2-Peer Netzwerk vorgestellt und miteinander verglichen.

6.1 Fluten

Fluten ist eine Möglichkeit, die Information an alle Knoten zu verbreiten. Hierbei schickt der Ausgangsknoten die Nachricht an alle seine direkten Nachbarn. Diese wiederum schicken die Nachricht an alle ihre Nachbarknoten weiter. Der Algorithmus endet, sobald der letzte Knoten alle seine Nachbarn benachrichtigt hat. Jeder Knoten flutet nur einmal, auch wenn er die Nachricht später erneut bekommt.

Dieser Ansatz stellt sicher, dass in einem stabilen Netzwerk alle Knoten die Information erhalten. Allerdings ist der Nachrichtenaufwand nicht optimal. Da jeder Knoten die Nachricht an alle seine Nachbarn schickt, erhalten alle Knoten die Information von sämtlichen Knoten, deren Nachbarn sie sind.

In einem typischen Peer-2-Peer Netzwerk hat jeder Knoten $O(\log n)$ Nachbarn. Dementsprechend erhält er die Information $O(\log n)$ mal. Dies hat zwar einen grossen Nachrichten Overhead zur Folge, führt aber dazu, dass ein Knoten auch dann noch die Information bekommt, wenn einzelne Nachrichten verloren gehen.

6.2 Broadcast Tree

In einem Broadcast Tree wird die Nachricht entlang eines Spannbaums über das Netzwerk geschickt (siehe Abbildung 6.1). Der Ausgangsknoten ist dabei die Wurzel des Baums. Jeder Knoten schickt, nachdem er die Nachricht erhalten hat, selbige an alle seine Kinder weiter. Da in einem Spannbaum jeder Knoten

genau einen Vorfahr hat, erhält in diesem Verfahren jeder Knoten die Nachricht genau einmal.

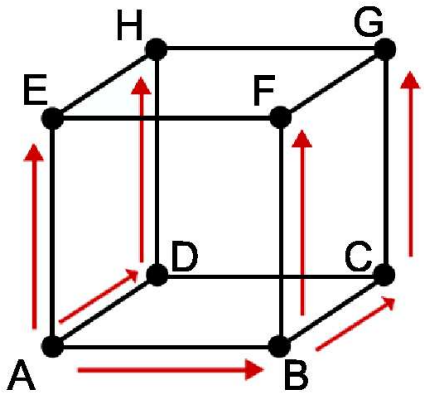

Abbildung 6.1: Ablauf eines Broadcast Tree Algorithmus in einem Netzwerk mit acht Knoten. Die Netzwerktopologie ist ein dreidimensionaler Hypercube. Ausgehend von Knoten A wird die Nachricht entlang den roten Pfeile an alle Knoten verschickt. Insgesamt wird die Nachricht siebenmal verschickt: dreimal von A, zweimal von B und je einmal von C und D.

Um allerdings einen Spannbaum erstellen zu können, müssen die Knoten Kenntnis von der Netzwerkstruktur haben. Wenn ein Knoten eine Nachricht von einem anderen Knoten erhält, muss er wissen, an welche Knoten er diese Nachricht weiter schicken soll. Entweder wird ihm dieses Wissen mit der Nachricht übermittelt oder er kann aufgrund der Identität des vorangegangenen Knotens ausrechnen, wem er die Nachricht weiter senden muss.

Der Vorteil der Informationsverteilung über einen Broadcast Tree ist der optimale Nachrichtenaufwand. Es werden insgesamt $n-1$ Nachrichten verschickt. Da genau $n-1$ Knoten informiert werden müssen, kann es keinen Algorithmus mit geringerem Nachrichtenaufwand geben. Allerdings ergibt sich aus dem minimalen Nachrichtenaufwand auch ein grosser Nachteil dieses Algorithmus. Dieser ist die geringe Stabilität. Wenn ein Knoten ausfällt oder eine Nachricht verloren geht, erhalten alle im Baum darunter gelegenen Knoten die Nachricht nicht. Abhängig von der Position des Ausfalls im Spannbaum werden so unter Umständen grosse Teile des Netzwerks nicht informiert.

6.3 Gossiping

Bei einem Gossiping Algorithmus wird die Nachricht nach dem Zufallsprinzip verbreitet. Wenn ein Knoten die zu verteilende Information bekommt, ermittelt er einen zufälligen Kommunikationspartner und schickt diesem die Nachricht weiter. Für das Gelingen eines Gossiping Algorithmus ist es entscheidend, dass jeder Knoten genügend Nachrichten verschickt, so dass mit hoher Wahrscheinlichkeit alle Knoten im Netzwerk informiert werden. Ausserdem müssen die Knoten ihre Kommunikationspartner unabhängig und mit gleichmässiger

Wahrscheinlichkeit wählen. In einem dynamischen Peer-2-Peer Netzwerk stellt dies eine nicht-triviale Aufgabe dar.

Da die Information zufällig verteilt wird, muss eine relativ grosse Anzahl von Nachrichten verschickt werden, um mit hoher Wahrscheinlichkeit jeden Knoten mindestens einmal zu erreichen. Ein einzelner Knoten muss die Nachricht $O(\log n)$ mal abschicken. Hinzu kommt noch der Nachrichtenaufwand, der durch die langen Wege in einem P2P System entsteht.

Ein Problem bei Gossiping in einem Peer-2-Peer Netzwerk kann die Nachrichtenlast auf einzelnen Verbindungen sein. Da die verschickten Nachrichten an zufällige Knoten im Netzwerk gesandt werden, kann es vorkommen, dass einzelne Verbindungen übermässig stark beansprucht werden. In Grenzfällen kann es dadurch zu einer Überlastung der Verbindung und zu Verzögerungen oder gar dem Verlust von Nachrichten kommen. Dies ist natürlich unerwünscht und es bleibt zu analysieren, wie hoch die Wahrscheinlichkeit einer kritischen Last ist.

6.4 Nachrichtenaufwand

Da in einem Broadcast Tree jeder Knoten die Information genau einmal zugesandt bekommt, ist der Nachrichtenaufwand $n - 1$. Dies entspricht einem optimalen Nachrichtenaufwand und kann nicht verbessert werden.

Da während dem Ablauf von Fluten jeder Knoten einmal alle seine Nachbarn benachrichtigt, werden $O(n \log n)$ Nachrichten versandt, wenn jeder Knoten $O(\log n)$ Nachbarn hat.

Wird der in Sektion 3.4.1 vorgestellte Push Algorithmus direkt in einem P2P System umgesetzt, entsteht ein Nachrichtenaufwand von $O(n \log^2 n)$. Also werden deutlich mehr Nachrichten als beim Broadcast Tree verschickt und immerhin um Faktor $\log n$ mehr als bei Fluten.

Das in dieser Arbeit vorgestellte Zwei-Phasen-Verfahren benötigt in der ersten Phase $O(n \log n)$ Nachrichten. Dies resultiert daraus, dass mindestens $\frac{1}{10}$ der Knoten je zwei weitere Knoten benachrichtigt. Für jede Benachrichtigung entstehen $O(\log n)$ Nachrichten im Netz, da die Information über eben soviele Stationen geleitet werden muss. In der zweiten Phase, der lokalen Verteilung, werden $O(n \log n)$ Nachrichten aufgewandt. Insgesamt werden während den zwei Phasen also $O(n \log n)$ Nachrichten verschickt.

Der Broadcast Tree Algorithmus benötigt mit Abstand die wenigsten Nachrichten. Fluten und das Zwei-Phasen-Verfahren sind gleichauf während der Push Algorithmus in einer P2P Umgebung deutlich am meisten Nachrichten verschickt.

6.5 Zeitaufwand

Für die Abschätzung des Zeitaufwands wird ein synchrones Modell der Algorithmen betrachtet. Während einer Runde kann ein Knoten genau eine Nachricht an einen seiner Nachbarn senden. Es wird davon ausgegangen, dass für das Versenden einer Nachricht auch nicht mehr Zeit als eine Runde benötigt wird.

In einem typischen P2P Netzwerk beträgt die maximale Distanz zwischen zwei Peers $O(\log n)$. Mit einem geeigneten Spannbaum benötigt ein Broadcast

Tree Verfahren dann auch $O(\log n)$ Runden, um das gesamte Netzwerk zu informieren.

Während des Fluten Vorgangs benachrichtigt jeder informierte Knoten seine Nachbarn einzeln. Ein einzelner Knoten ist also während $O(\log n)$ Runden aktiv. Da der maximale Weg zwischen zwei Knoten über $O(\log n)$ Knoten geht, benötigt Fluten also $O(\log^2 n)$ Runden, um alle Knoten zu informieren.

Die erste Phase des in dieser Arbeit vorgestellten Zwei-Phasen-Verfahrens benötigt, wie in Kapitel 5 gezeigt, $O(\log^2 n)$ Runden um den verlangten fixen Anteil aller Knoten zu benachrichtigen.Die lokale Verteilung benötigt $O(\log n)$ Runden. Der gesamte Vorgang ist also nach $O(\log^2 n)$ Runden abgeschlossen.

Wird der Push Algorithmus aus Sektion 3.4.1 direkt in einem P2P Netzwerk angewandt benötigt er, wie das Zwei-Phasen-Verfahren, $O(\log^2 n)$ Runden.

Der Zeitaufwand des Broadcast Tree Algorithmus ist der geringste der vorgestellten Algorithmen. Fluten und die Gossiping Verfahren benötigen jeweils Faktor $\log n$ mehr Runden.

6.6 Robustheit

Von den vorgestellten Algorithmen ist der Broadcast Tree am anfälligsten für Ausfälle von Knoten und/oder Nachrichten. Wenn ein Knoten die Information nicht erhält, bleiben auch alle im Spannbaum unter ihm gelegenen Knoten uninformiert. Dies ist der Preis, der für den optimalen Nachrichtenaufwand dieses Algorithmus gezahlt werden muss.

Fluten schickt Nachrichten an alle seine Nachbarn. Damit ein Knoten die Nachricht nicht erhält, müssen mindestens alle seine Nachbarn ausfallen. Das sind $O(\log n)$ Knoten. In einem balancierten Kademlia System führen $O(k \log \frac{n}{k})$ Verbindungen in ein Subnetz mit k Knoten hinein, und ebenso viele führen hinaus. Damit also k Knoten nicht erreicht werden, müssen $O(k \log \frac{n}{k})$ Knoten oder Nachrichten ausfallen. Fluten ist wesentlich robuster gegenüber Ausfällen als das Broadcast Tree Verfahren.

Ein Gossiping Algorithmus in P2P schickt seine Nachrichten ebenfalls an seine Nachbarn. Diese leiten die Mitteilung dann gegebenenfalls weiter. Dadurch ist ein Knoten, ähnlich wie bei Fluten, abhängig von der Stabilität seiner Nachbarn. Wenn diese ausfallen, kann die Nachricht nicht weiter verbreitet werden.

Während Gossiping und Fluten beide sehr robust gegenüber Ausfällen von Knoten und Nachrichtenverlust sind, reagiert der Broadcast Tree Algorithmus sehr empfindlich auf den Ausfall von nur wenigen Knoten. Der Mehraufwand an Nachrichten und Zeit, den Gossiping und Fluten benötigen, ist der Preis für das Plus an Stabilität.

6.7 Simplizität

Fluten ist der simpelste der vorgestellten Algorithmen. Jeder Knoten schickt eine erhaltene Nachrichten an alle seine Nachbarn weiter. Er muss lediglich darauf achten, die gleiche Information bei erneutem Erhalten nicht noch einmal weiterzuschicken, damit kein unnötiger Mehraufwand an Nachrichten entsteht.

Der Push Algorithmus ist etwas umständlicher, da ein Knoten vor dem Verschicken einer Nachricht zusätzlich einen zufälligen Adressaten ermitteln muss.

Das Verfahren ist aber für jeden Knoten gleich und symmetrisch.

Das vorgestellte Zwei-Phasen-Verfahren ist etwas komplizierter als der Push Algorithmus, da ein Knoten unterscheiden muss, in welcher Phase er informiert wurde. Je nach dem muss er die Nachricht an zufällige Knoten oder nur an seine Nachbarn verteilen. Auch muss ein Knoten merken wenn er eine Nachricht zum wiederholten Mal erhält.

Broadcast Tree ist das komplizierteste der vorgestellten Verfahren. Jeder Knoten, der die Nachricht erhält, muss wissen, welche Knoten im verwendeten Spannbaum seine Kinder sind. Gerade in einem dynamischen Peer-to-Peer Netzwerk kann dies kompliziert werden.

6.8 Diskussion

Broadcast Tree ist das einzige der vorgestellten Verfahren, welches mit der minimalen Anzahl Nachrichten auskommt. Auch ist der Algorithmus sehr schnell. Allerdings ist der Preis der hierfür gezahlt werden muss mangelnde Robustheit und ein kompliziertes Verfahren zur Ermittlung der Adressaten.

Gossiping und Fluten sind beides Verfahren, welche deutlich mehr Nachrichten verschicken, als minimal nötig wären. Aus diesem Mehraufwand folgt allerdings eine grössere Robustheit, was klar der Vorteil dieser Vorgehen ist. Das Zwei-Phasen-Verfahren benötigt um Faktor $O(\log n)$ weniger Nachrichten als eine an P2P angepasste Version des Push Algorithmus. Allerdings ist das Zwei-Phasen-Verfahren trotzdem weder in Bezug auf Nachrichtenaufwand noch in Bezug auf Zeitaufwand besser als Fluten. Da Gossiping in P2P seine Nachrichten wie Fluten auch über die direkten Nachbarn eines Knoten verschicken muss, wird durch Gossiping auch keine grössere Robustheit erreicht als mit Fluten. Da aber Fluten in der praktischen Umsetzung um einiges simpler ist als Gossiping, gibt es keine stichhaltigen Argumente, warum man den vorgestellten Algorithmus in einer P2P Umgebung einsetzen würde.

Kapitel 7

Fazit

In dieser Arbeit wird das Konzept von Gossiping untersucht. Die einem Gossiping Algorithmus zugrunde liegenden Prinzipien werden vorgestellt und es wird besprochen, wie Gossiping in einem Peer-to-Peer Netzwerk angewandt werden kann. Darauf aufbauend wird ein Gossiping Algorithmus für ein Peer-to-Peer Netzwerk vorgeschlagen und untersucht. Es wird bewiesen, dass mit hoher Wahrscheinlichkeit mit diesem Algorithmus alle Knoten erreicht werden.

Das vorgestellte Zwei Phasen Verfahren wird mit anderen Broadcast Algorithmen verglichen. Im Vergleich zum Broadcast Tree erweist sich das Zwei Phasen Verfahren zumindest als robuster. Es stellt sich jedoch heraus, dass Fluten im Aspekt von Nachrichten- und Zeitaufwand ebenbürtig ist und auch nicht weniger robust abläuft. Fluten ist im Hinblick auf die Simplizität der Verteilung Gossiping sogar leicht überlegen.

7.1 Gossiping und Fluten

Dass Fluten in einem P2P Netzwerk Gossiping mindestens ebenbürtig ist rührt daher, dass jeder Knoten $O(\log n)$ Nachbarn hat und nur mit diesen direkt kommunizieren kann. Für Fluten heisst dies, dass ein Knoten nur $O(\log n)$ Nachrichten an seine Nachbarn schicken muss. Für Gossiping bedeutet dies aber, dass die Kommunikation mit zufälligen Knoten des Netzwerks umständlicher wird. Um einen zufälligen Knoten im Netzwerk zu erreichen werden $O(\log n)$ Nachrichten benötigt. Daraus resultiert, dass der Nachrichtenaufwand von Gossiping gleich gross ist wie der von Fluten.

Um den Nachrichtenaufwand von Gossiping zu minimieren, könnte ein Gossiping Algorithmus entwickelt werden, welcher die Information an zufällig gewählte Nachbarn schickt. Dadurch werden die Nachrichten nicht entlang langer Wege durch das Netzwerk geschickt. In Sektion 3.3.3 wurde jedoch gezeigt, dass jeder Knoten mindestens $O(\log n)$ Nachrichten verschicken muss, damit am Ende das ganze Netzwerk informiert ist. Da ein einzelner Knoten in einer P2P Umgebung gerade $(O \log n)$ Nachbarn hat, käme ein solcher Gossiping Algorithmus einem Fluten gleich.

Man kann Fluten in P2P in einer gewissen Weise sogar als Gossiping betrachten. Die Anordnung der Peers geschieht beim Aufbau des Systems meistens zufällig, oder zumindest werden die Peers in zufälliger Reihenfolge in das

System eingefügt. Die Kommunikationspartner werden also bei der Bildung des Netzes gewählt, statt direkt während der Verteilung einer Information. Umgekehrt kann ein Gossiping Algorithmus in einem Netzwerk, in dem jeder jeden kennt, als Aufbau eines Subgraphen betrachtet werden. Die Nachricht wird dann entlang den Kanten dieses Subgraphen verschickt, was einem Fluten dieses desselben entspricht.

7.2 Ausblick

In dieser Diplomarbeit wurde immer von P2P Netzwerken ausgegangen, in denen ein Knoten $O(\log n)$ Nachbarn hat. Es sind aber durchaus Netzwerke denkbar, in denen ein Knoten mehr oder auch weniger Nachbarn hat. Wie sich Gossiping in solchen Netzen verhält, wäre ein möglicher Ansatz für weitere Forschungen.

Sollte der Ansatz von Gossiping in P2P Netzwerken weiter verfolgt werden, dann ist zu untersuchen, wie sich die Nachrichtenlast auf die Verbindungen verteilt. Da im Verlauf des Gossiping Algorithmus viele Nachrichten verschickt werden, kann es vorkommen, dass es zu Anhäufungen von vielen Nachrichten auf einzelnen Verbindungen kommt. Dies könnte Stauungen und Überlastung der Verbindungen nach sich ziehen.

Will man, wie in Sektion 4.3.1 beschrieben, Distributed Hash Tables zur Ermittlung zufälliger Adressaten eines Gerüchts benützen, so muss untersucht werden, wie sich die Verteilung der Information bei unbalancierten Systemen verhält. Es ist zu klären, wie unbalanciert das System maximal sein darf, damit trotzdem mit hoher Wahrscheinlichkeit alle Knoten des Netzwerks mit Gossiping erreicht werden.

Literaturverzeichnis

[1] Intel® philanthropic peer-to-peer program. http://www.intel.com/cure/.

[2] KaZaa. http://www.kazaa.com.

[3] Napster. http://www.napster.com.

[4] C. Hauser W. Irish J. Larson S. Shenker H. Sturgis D. Swinehart A. Demers, D. Greene and D. Terry. Epidemic algorithms for replicated database maintenance, 1987. In Proceedings of the Sixth Symposium on Principles of Distributed Computing, pages 1-12, August 1987.

[5] Keno Albrecht, Ruedi Arnold, Michael Gähwiler, and Roger Wattenhofer. Join and leave in peer-to-peer systems: The DASIS approach. Technical report, ETH Zürich, November 2003.

[6] Norman T. J. Bailey. *the Mathematical Theory of Infectious Diseases and its Applications (second edition).* Hafner Press, 1975.

[7] Andrew Barbour and Denis Mollison. Epidemics and random graphs. In *Stochastic Processes in Epidemic Theory*, pages 86–89. Springer Verlag, 1988.

[8] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observation. *Ann. Math. Stat.*, 23:493–509, 1952.

[9] Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vocking. Randomized rumor spreading. In *IEEE Symposium on Foundations of Computer Science*, pages 565–574, 2000.

[10] Ruedi Arnold Keno Albrecht and Roger Wattenhofer. Clippee: A large-scale client/peer system. In the Proceedings of the International Workshop on Large-Scale Group Communication, held in conjunction with the 22nd Symposium on Reliable Distributed Systems (SRDS), Florence, Italy, October 2003.

[11] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric, 2002.

[12] Boris Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47:213–223, February 1987.

[13] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, pages 149–160. ACM Press, 2001.