Stochastic Analyses of Dynamic Computer Processes

Gopal Pandurangan

Ph.D. Dissertation

Department of Computer Science
Brown University
Providence, Rhode Island

May 2002

Stochastic Analyses of Dynamic Computer Processes

by

Gopal Pandurangan

B. Tech. (Computer Science), Indian Institute of Technology at Madras, 1994

M. S. (Computer Science), State University of New York at Albany, 1997

Sc. M. (Computer Science), Brown University, 1999

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

May, 2002

This dissertation by Gopal Pandurangan is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Professor Eli Upfal, Director

Recommended to the Graduate Council

Date _____ _____

Professor Franco Preparata, Reader

Date _____ _____

Professor John Savage, Reader

Approved by the Graduate Council

Date _____ _____

Dean of the Graduate School and Research

Vita

Name Gopal Pandurangan

Born July 20th 1971, Chennai, India.

Education Jawahar Vidyalaya Senior Secondary School, Chennai, India
High School Certificate, 1988.

Indian Institute of Technology at Madras, Chennai, India
Bachelor of Technology in Computer Science, June 1994.

State University of New York at Albany, Albany, NY
Masters in Computer Science, May 1997.

Brown University, Providence, RI
Sc. M. in Computer Science, May 1999.

Brown University, Providence, RI
Ph. D. in Computer Science, May 2002.

Honors All India First in Mathematics, Class XII public examination
Central Board of Secondary Education, India, 1988.

National Talent Search Scholarship, Government of India, 1986-1994.

Junior Research Fellowship, Institute of Mathematical Sciences, Chennai, India, 1994-1995.

University Fellowship, Brown University, 1997-1998.

Outstanding Graduate Student Award, Sigma Xi, 2001.

அருமை உடைத்தென்று அசாவாமை வேண்டும்
பெருமை முயற்சி தரும்
Say not, 'tis hard', in weak, desponding hour
For strenuous effort gives prevailing power.

— Thirukural (a Tamil Classic), verse 611
by Thiruvalluvar, *circa* 100 AD
(Translation by Rev. Dr. G.U. Pope)

Abstract

Stochastic processes can model a variety of *dynamic* computer phenomena such as traffic in communication networks, input for packet routing or caching or load balancing, growth of the World Wide Web, and evolving structure of dynamic networks. Stochastic analysis enables us to characterize properties of such processes that dynamically change with time.
In this thesis, we present four results in different settings in which stochastic processes serve as natural models for dynamic computer phenomena that arise in practical applications. The models are useful not only in understanding these phenomena but also in designing and analyzing practical algorithms for associated problems.

The results are:

1. *Communication Networks:* We address the problem of evaluating Quality-of-Service (QoS) properties in statistically multiplexed communication networks fed by *bursty* sources. Using a stochastic model for bursty sources, we give efficient *Monte-Carlo* algorithms for estimating the failure probability of an arbitrary topology network under static and dynamic settings.

2. *Peer-to-Peer Networks*: We address the fundamental problem of building Peer-to-Peer (P2P) networks with good topological properties. We present a practical and scalable *distributed* protocol for building P2P networks and prove under a reasonable stochastic model that it results in *connected* networks of *constant* degree and *logarithmic* diameter.

3. *Online Computation*: We propose a novel approach for measuring online performance based on the *characteristics* of the input sequence; this is fundamentally different from the standard competitive analysis of online computation. Assuming a very general stochastic model for the input sequence, we present bounds between *entropy* of the input and the performance of the best online algorithm for classical online problems such as prefetching, caching, and load balancing.

4. *Web Models*: We develop and analyze new stochastic models for the Web graph which capture a global property of the Web: the *PageRank* distribution. Our models explain the power law property of the PageRank distribution while remaining faithful to the properties of the previously studied degree distributions.

To my family

and

To my grandparents Dr. V.B. Venugopal and Mrs. Visalakshi Venugopal, in memoriam

Acknowledgments

நன்றி மறப்பது நன்றன்று நன்றல்லது
அன்றே மறப்பது நன்று
'Tis never good to let the thought of good things done thee pass away;
Of things not good, 'tis good to rid thy memory that very day.

- Thirukural, verse 108

My Ph.D. pursuit has been a fantastic journey. I couldn't have successfully reached the destination without help from many professors, teachers, colleagues, friends, family, and well-wishers. At the risk of omitting names, I will thank those who have helped me in this rewarding journey.

Quite simply, this thesis wouldn't have been possible without Professor Eli Upfal, my advisor. It was coincidence that we joined Brown University almost at the same time. I was lucky to find an advisor in Eli who not only matched my research interests but also was ideal to collaborate. I have benefited a lot from our innumerable discussions (almost on a daily basis - I could walk any time into his office and start a discussion!) in the last four years. The result is that almost all of my papers at Brown are co-authored with Eli. Eli has a fantastic knowledge and depth in many areas; he has a tremendous knack of making even the most difficult concepts appear clear and simple. His approach and style has influenced me very much. I thank him for his continuous enthusiastic encouragement and for supporting me as a research assistant for many semesters and summers.

Brown has been a most wonderful and friendly place for me. The atmosphere in the department can hardly be better. I am very thankful to the great faculty we have; the excellent courses I have taken here are testimonies to that. I am especially grateful to Professors Franco Preparata and John Savage who served on my thesis committee. Professor Preparata inspired me to work in computational biology; his course on this exciting field was a great motivation. He motivated me to work on the partial digest problem; this turned into a journal paper. I am very grateful to Professor Savage for his continuous guidance and encouragement at various crucial moments of my Ph.D. I cannot forget his insightful advice after the programming exam in my first year, nor I can forget his enthusiastic support during my faculty job search. I am privileged to be amidst such wonderful great people.

I thank Dr. Prabhakar Raghavan for giving me the rewarding opportunity of working with him at Verity in the summer of 2001. Some of the results in this thesis were obtained in collaboration

with him. He is a fantastic researcher and I enjoyed working with him.

My Ph.D. experience wouldn't have been so enjoyable if not for the friendship of many fellow graduate students. I was fortunate to have the friendship of Rahul Bose in my first year at Brown; his support and companionship was invaluable in getting through the difficult early semesters. I will always fondly recall the countless chess games we played in 1998. Dnyanesh Pawaskar has been a wonderful friend and roommate. His support and companionship was also invaluable in the early years. I am also very thankful to Vasanth Kothnur (I nicknamed him "Vasanteh" !) for his innumerable favors; his energy and dynamism made possible many dinners, movies, cricket games, and weekend trips. I really enjoyed these fun get-togethers with my friends - Nitin and Deepali Bhate, Kedar Hardikar, Anand Padmanabhan, Ram Krishnamurthy, Ashok Rajamani, and Suvarna Sathe. I also thank the following department colleagues for their time and support: Aris Anagnostopoulos, Costas Busch, Milos Hauskrecht, Olga Karpenko, Laurent Michel, Luis Ortiz, Prabhat, Manos Renieris, Srikanta Tirthapura, Robert Velisar, and Joel Young.

I am very fortunate to know Professor S.S. Ravi from SUNY, Albany. He has been a constant source of support, friendship, and guidance ever since I came to USA. I am very grateful for his numerous favors. I am also fortunate to know Ramakrishna Earasi, a long-time well-wisher and a family friend.

This thesis wouldn't have been possible without the support of my family in India: my parents,my aunt and my two sisters. They have always believed in me; they allowed me the luxury of focusing only on my goal here. They have endured, as I have, the long separation with grace and courage.

I am indebted to my wife Sumithra for everything I have achieved in the last two years. Her love and friendship is incomparable. She has endured long hours of loneliness for the sake of my research and has been most accommodating and supportive of my effort. My life can't be complete without her. I dedicate this thesis to her and to my family in India and to my late grandparents who set me on the right path when I was young.

Contents

⋆ Parts of this thesis have been previously published in the following journals/conferences with the specified co-authors: ACM Symposium on the Theory of Computing 1999 (with Eli Upfal), Journal of Interconnection Networks (with Eli Upfal), ACM-SIAM Symposium on Discrete Algorithms 2001 (with Eli Upfal), IEEE Symposium on the Foundations of Computer Science 2001 (with Prabhakar Raghavan and Eli Upfal), and International Computing and Combinatorics Conference 2002 (with Prabhakar Raghavan and Eli Upfal).

List of Figures

Chapter 1

Introduction

Traditional algorithmic analyses have focused mainly on static computation problems where the input is known before the start of the computation and the goal is to minimize the time till termination with a correct output. Many important processes in today's computing are *dynamic processes*, whereby input is *continuously* injected to the system, and the algorithm is measured by its long term or steady state performance. Examples of dynamic computer processes include traffic in communication networks, protocols for caching, prefetching, load balancing, or packet routing, growth of real-world networks such as the World Wide Web, and evolving structure of dynamic networks such as Peer-to-Peer networks or Ad hoc mobile networks.

The general theme of this thesis is using *stochastic processes* to model dynamic computer phenomena. They are *probabilistic* models. Mathematically, they are simply an indexed family of random variables representing or characterizing the input. They are ideal to model phenomena that dynamically change with time. Stochastic models are used extensively in physics, biology, finance, engineering and social sciences. Brownian motion, Poisson process, branching process and diffusion process are some well-known examples of stochastic processes in the above areas. For example, Brownian motion is used to model the price of a stock or the motion of small particles suspended in a liquid; Poisson process is used to model the arrival of customers in a queue or the emission of radioactive particles.

In this thesis, we present four results in different settings in which stochastic processes serve as natural models for dynamic computer phenomena. Our motivation is to illustrate the usefulness of stochastic models and analysis in diverse application areas ranging from Peer-to-Peer networks to Web modeling. Stochastic analysis is useful not only in understanding the dynamic phenomena in these settings but also in designing and analyzing practical algorithms/schemes for associated problems. Each result also illustrates a few typical themes in stochastic analysis. One of the themes in our analyses is handling *dependencies* among the random variables. When the random variables are mutually independent analysis is easy. However, in many applications dependencies arise naturally. The general idea in these situations is to bound the dependence by some clever means so as to facilitate analysis.

Before going to an overview of our results, it will be illustrative to give an example of a dynamic computer phenomenon and a stochastic model for it. We consider the fascinating dynamic phenomenon of the evolution the World Wide Web graph. The Web graph is simply a directed graph where the nodes are the webpages and the hyperlinks between webpages form the directed edges in the natural manner. The following simple model for the Web graph was proposed by Barabasi *et.al* [11]:

1. *uniform growth*: Nodes and edges are added one at a time;
2. *preferential attachment*: The incoming node connects to an existing node with probability proportional to its total *degree*.

The above model was motivated by the following remarkable property of the Web graph: the degree distribution follows a *power law* whose exponent is a small constant irrespective of the size of the graph. The above model induces a stochastic graph process $\mathcal{G} = (G_t)_{t \geq 0}$ defined as follows:

- G_t is a graph on $\{v_i : 0 \leq i \leq t\}$;
- G_0 has one vertex v_0 with a self-loop;
- Given G_{t-1} we form G_t by adding the vertex v_t together with an edge between v_t and v_i where i is chosen randomly with probability

$$\Pr(i = s) = \frac{d_{G_{t-1}}(v_s)}{2t} \quad 0 \leq s \leq t-1$$

The above process can be analyzed to show that it captures the power-law property of the degree distribution of the Web. Thus, even a simple model as above gives insight into understanding a complex dynamic phenomenon as the growth of the Web.

1.1 Overview

We now present a brief overview of our results below. More details on motivation, analysis, and applications of the four results can be found in Chapters 3, 4, 5, and 6 respectively. Here we would like to restrict ourselves to giving a general idea of the stochastic processes that arise in our results. The results are ordered according to the increasing order of complexity of the corresponding stochastic models used: a simple discrete memoryless model (Chapter 3), a Poisson model (Chapter 4), a general stationary model (Chapter 5), and a non-stationary model (Chapter 6). (For a quick reference to the concepts and terms here the reader is referred to Chapter 2.)

Communication Networks: ([62, 61]) In the first setting, we model a communication request in a statistically multiplexed communication network by a simple stochastic source: an *on-off source*. Though simple, it captures the key feature of a *bursty* source, common in the Internet traffic. The problem is to evaluate Quality-of-service (QoS) parameters in an arbitrary communication network fed by many communication requests. Such problems are hard to analyze rigorously, and typically

ad-hoc estimates are used in practice. Two important assumptions facilitate our analysis here. The first is *independence* among communication requests, which enables us to use well-known techniques from probability theory concerning large deviations. Second is independence *between time steps* of a connection. This allows us to describe the source in a very simple way, abstracting out the notion of time.

Our main result, is an efficient Monte-Carlo algorithm for estimating the failure probability of a general network. Our method is particularly useful in a dynamic setting in which communication requests are dynamically added and eliminated from the system. The amortized cost in our solution of updating the estimate after each change is proportional to the fraction of links involved in the change rather than to the total number of links in the network.

Peer-to-Peer Networks: ([64]) We address the fundamental problem of building P2P networks with good topological properties. We use a standard stochastic model from queuing theory to model P2P networks: computers join the network independently according to a *Poisson* process with rate λ and the duration with which a computer stays connected is *exponentially* distributed with parameter μ. We present a simple *distributed local* protocol for building P2P networks and prove under the above model that it results in *connected* networks of *constant* degree and *logarithmic* diameter. An important feature of our protocol is that it operates without any global knowledge of all the nodes in the network. To our knowledge, this is the first such protocol with provable guarantees on connectivity and diameter under a realistic dynamic setting.

The technical contribution of our analysis is a non-trivial analysis of an evolving graph with nodes and edges arriving and leaving the network according to the protocol. The analysis is quite challenging because of the dependencies among the random variables which naturally arises in our analysis. However our model has two key properties which makes it tractable: *independence* between the peers and the *memorylessness* of the peer duration times.

Online Computation: ([63]) We model the input sequence (also called "request sequence") for an online problem as a stochastic process. We then analyze the performance of the best online algorithm (for different online problems) based on the characteristics of the input sequence. This is a very different approach compared to the standard competitive analysis of online algorithms. We use *entropy* of the stochastic input to study online performance. Entropy is a measure of the *uncertainty* of a stochastic process. Intuitively, we expect online algorithms to perform well on highly predictive sequences (low entropy), rather than sequences with little pattern or high randomness (high entropy). We study the relation between entropy of the request sequence and the performance of the best online algorithm for three classic online problems: list accessing, caching and prefetching. Our approach is motivated by practical applications (such as partitioning a malleable cache) where we need to quantify the resources a system needs to allocate for a given online process.

In our stochastic model, there can be arbitrary dependencies among the random variables. However, our assumptions of *stationarity* and *ergodicity* allows us to use powerful results from information theory.

Web Models: ([65]) We develop and analyze (both analytically and experimentally) models for the Web graph. Our models explain the PageRank distribution (a "global" property of the Web which is used in Google to rank Web pages) while incorporating the previously studied degree distributions (a "local" property). To our knowledge this represents the first modeling of the Web that goes beyond fitting degree distributions on the Web.

The stochastic graph process induced by our models do not have the properties of stationarity or independent increments. The models are difficult to analyze rigorously; however we do manage to give heuristic arguments which give good insight into the main feature we are interested in modeling: the PageRank and degree distributions. We also perform experimental simulations to support the theoretical analysis.

1.2 Organization

In chapter 2, we give a brief review of key results from probability theory, stochastic processes, and information theory used in our analysis. The next four chapters (3, 4, 5, and 6) deal in detail the four results we outlined briefly above. Each chapter starts with a detailed motivation for the problem being addressed. Also a clear-cut rationale is presented for the stochastic modeling and analysis that follows. Each chapter also discusses related work and pointers for further work wherever possible. We end the thesis with a brief epilogue.

Chapter 2

Stochastic Processes and Probabilistic Techniques

In this chapter, we briefly review basic concepts from probability theory, stochastic processes, and information theory. The first section lists useful inequalities from probability theory. The next section gives a brief introduction to martingales which are useful in handling dependent random variables. The next two sections give a brief introduction to stochastic processes (in particular Poisson processes) and information theory. This chapter is intended mainly as a quick reference for other chapters in the thesis; readers familiar with this material can skip to a subsequent chapter of their interest. For detailed expositions and proofs we refer to standard texts mentioned.

2.1 Inequalities from Probability Theory

We review basic inequalities which are used frequently in our analysis. Especially useful of these are the "large-deviation" inequalities which bound the probability that a certain random variable takes a values far away from the mean. For proofs and more detailed expositions we refer to [56] and [3].

A very useful inequality is *Boole's* inequality, also called as the *union bound*. Let $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ be arbitrary events. Then

$$\Pr(\cup_{i=1}^{n} \mathcal{E}_i) \leq \sum_i \Pr(\mathcal{E}_i) \tag{2.1}$$

The union bound is often used to bound the sum of probabilities of a set of events - no special assumption on the set of events (such as independence) is needed.

Another useful inequality is the following special case of the *Boole-Bonferroni* inequality which gives a lower bound on the probability of a union of events.

$$\Pr(\cup_{i=1}^{n} \mathcal{E}_i) \geq \sum_i \Pr(\mathcal{E}_i) - \sum_{i<j} \Pr(\mathcal{E}_i \cap \mathcal{E}_j) \tag{2.2}$$

A common theme in probabilistic analysis is to show that a random variable is not far away from its mean, with large probability. Let X be a random variable and let $\mu = E[X]$ and $\sigma^2 = Var(X)$. A simple and useful inequality called *Markov's* inequality which holds for any positive random variable is

$$\Pr(X \geq t) \leq \frac{\mu}{t} \tag{2.3}$$

for any $t > 0$.

Another useful inequality due to *Chebyshev's* gives a stronger bound for any random variable with a finite variance:

$$\Pr(|X - \mu| \geq t) \leq \frac{\sigma^2}{t^2} \tag{2.4}$$

for any $t > 0$.

We will often consider random variables that are indicator functions of some events; such random variables are called *indicator* or *zero-one* random variables. Let $X_1, \ldots, X_n$ be *independent* 0-1 random variables such that $\Pr(X_i = 1) = p_i$, where $0 < p_i < 1$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X] = \sum_{i=1}^{n} p_i$. For any $\delta > 0$ we have the following large deviation bounds (also known as *Chernoff bounds*):

$$\Pr(X > (1+\delta)\mu) < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} \tag{2.5}$$

For $0 < \delta < 1$ we have the following bounds:

$$\Pr(X < (1-\delta)\mu) \leq e^{-\mu\delta^2/2} \tag{2.6}$$

$$\Pr(X > (1+\delta)\mu) \leq e^{-\mu\delta^2/3} \tag{2.7}$$

Note that if $p_i = p$ for all i, then X is simply a binomially distributed random variable with mean np. Thus the above bounds can be interpreted as bounding the tail probabilities of the binomial distribution.

We also use Chernoff-like bounds involving distributions other than the sums of indicator random variables. Let P have a Poisson distribution with mean μ. Let $\epsilon > 0$. Then we have:

$$\Pr(P \leq \mu(1-\epsilon)) \leq e^{-\epsilon^2\mu/2} \tag{2.8}$$

$$\Pr(P \geq \mu(1+\epsilon)) \leq \left[e^{\epsilon}(1+\epsilon)^{-(1+\epsilon)}\right]^{\mu} \tag{2.9}$$

We use Chernoff bounds extensively to bound the probability that a random variable is not far away from the mean. The zero-one estimator theorem ([46]) used in chapter 3 is a straightforward consequence of the Chernoff bound.

2.1.1 Martingale Inequalities

Martingales are useful in handling sums of random variables which are not totally independent. (Note that the Chernoff bounds given above assume that the random variables are all independent.) Our discussion on martingales will be brief and focussed. A good reference for martingales and their algorithmic applications is [56].

A sequence of random variables $X_0, X_1, \ldots$, is said to be a *martingale sequence* if for all $i > 0$, $E[X_i|X_0, \ldots, X_{i-1}] = X_{i-1}$. An immediate consequence of this definition is that $E[X_i] = E[X_0]$. This property is very useful in obtaining Chernoff-like bounds even if the random variables are dependent provided they have the property of *bounded differences*, made precise below. Let $X_0, X_1, \ldots$ be a martingale sequence such that for each k,

$$|X_k - X_{k-1}| \leq c_k$$

where c_k may depend on k. Then *Azuma's inequality* gives for all $t \geq 0$ and any $\lambda > 0$,

$$\Pr(|X_t - X_0| \geq \lambda) \leq 2e^{-\frac{\lambda^2}{2\sum_{k=1}^{t} c_k^2}} \tag{2.10}$$

Thus Azuma's inequality gives a Chernoff-like bound for martingale random variables. One of the reasons why martingales (and the method of bounded differences) prove so useful in stochastic analysis is because we can construct a martingale sequence from any random variable. Let $Z_1, Z_2, \ldots, Z_n$ be *any* sequence of random variables (they can be dependent on each other in an arbitrary fashion) and let X be any random variable. Define the random variable $X_i = E[X|Z_1, \ldots, Z_i]$, i.e., the conditional expectation of X conditioned on the variables Z_1 to Z_i. Then $X_0 = E[X], X_1, \ldots, X_n$ form a martingale sequence. Martingales obtained in this manner are sometimes referred to as *Doob* martingales.

A particular case of a Doob martingale in the context of random graphs is known as *(vertex) exposure martingale* defined as follows. Let $G = (V, E)$ be a random graph. For each edge $e_j \in E$, define the indicator random variable I_j which takes the value of 1 when the edge e_j is present in G, and has value zero otherwise. In general, these indicator random variables can be dependent on each other. Let $Y = f(I_1, I_2, \ldots I_{|E|})$ be any real-valued measurable function. For $1 \leq i \leq |V|$, let E_i be the set of all possible edges with both endpoints in $\{1, \ldots, i\}$. Define Y_i as the (conditional) expectation of Y, conditioned by the knowledge of the indicator variables I_j for all $j \in E_i$. Then the exposure martingale sequence is defined as $Y_0 = E(Y), Y_1, \ldots, Y_{|V|} = Y$.

In the above setting, if $|Y_i - Y_{i-1}| \leq c_i$, for each i, then the method of bounded differences gives

$$\Pr(|Y - E(Y)| \geq \lambda) \leq 2e^{-\frac{\lambda^2}{2\sum_{i=1}^{|V|} c_i^2}} \tag{2.11}$$

A martingale is a special case of a very general probabilistic concept known as a stochastic process. For the sake of completeness we introduce the basic theory concerning stochastic processes with emphasis on stationary processes. Our main references for the next section are [69] and [45].

2.2 Stochastic Processes

A stochastic process $\{X_t, t \in T\}$ is a family of random variables. The variable t is often interpreted as time, and hence X_t represents the *state* of the process at time t. The set of possible values which the random variables $X_t, t \in T$ may assume is called the *state space* of the process. For example, X_t may represent the number of customers in a queue at time t or the page request in a caching system at time t or the position of a particle at time t. The set T is called the *index set* of the stochastic process. If T is a countable set, then the stochastic process is said to be a *discrete-time* process. If T is an open or a closed interval of the real line, then the stochastic process is said to be a *continuous time* process. Both types of processes are encountered in this thesis: chapter 5 deals with a discrete-time process while the graph process of chapter 4 is a continuous-time process.

An important special class of stochastic processes is stationary processes. A *stationary* process is one whose probabilistic description is time invariant. More formally,

Definition 2.2.1 *A discrete-time stochastic process* $\{X_t\}, t \in \{0, 1, \dots\}$ *is* stationary *if the joint distribution of any subset of the sequence of random variables is invariant with respect to shifts in the time index, i.e.,*

$$\Pr\{X_1 = x_1, X_2 = x_2, \dots, X_n = x_n\} =$$
$$\Pr\{X_{1+t} = x_1, X_{2+t} = x_2, \dots, X_{n+t} = x_n\}$$

for every shift t *and for all* $x_1, x_2, \dots, x_n \in \mathcal{H}$.

Analogously, a continuous-time stochastic process $\{X_t\}$ is said to be stationary if $X_{t_2+s} - X_{t_1+s}$ has the same distribution as $X_{t_2} - X_{t_1}$ for all $t_1, t_2 \in T$ and $s > 0$.

Another important special property that many stochastic processes used in applications possess is that of *independent increments*: a continuous-time process is said to have independent increments if for all choices of $t_0 < t_1 < \dots t_n$, the n random variables

$$X_{t_1} - X_{t_0}, X_{t_2} - X_{t_1}, \dots, X_{t_n} - X_{t_{n-1}}$$

are independent. Well-known examples of processes with stationary independent increments are the *Poisson* process and the *Brownian motion (Wiener)* process. We now describe in some detail the Poisson process, an important process which arises in many modeling applications.

2.2.1 Poisson Process

A Poisson process is a special type of *counting* process. A stochastic process $\{N_t, t \geq 0\}$ is said to be a counting process if N_t represents the total number of events which have occurred up to time t. Standard examples are the number of customer arrivals up to t or the number of radioactive particles emitted up to t.

Definition 2.2.2 *The counting process* $\{N_t, t \geq 0\}$ *is said to be a* Poisson Process *if*

1. $N_0 = 0$;
2. $\{N_t, t \geq 0\}$ *has independent increments;*
3. *The number of events in any interval of length* t *has a Poisson distribution with mean* λt. *That is, for all* $s, t \geq 0$,

$$\Pr(N_{t+s} - N_s = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!}, n \geq 0$$

From 3, it follows that $E[N_t] = \lambda t$ and λ is called the *rate of the process.*

Another equivalent definition which is sometimes useful in proving whether a certain process is Poisson is the following.

Definition 2.2.3 $\{N_t, t \geq 0\}$ *is a Poisson process if*

1. $N_0 = 0$;
2. $\{N_t, t \geq 0\}$ *has stationary independent increments;*
3. $\Pr(N_t \geq 2) = o(t)$;
4. $\Pr(N_t = 1) = \lambda t + o(t)$

There is yet another way of characterizing a Poisson process in terms of the *sequence of interarrival times.*

Proposition 2.2.1 *Consider a Poisson process with rate* λ. *Let* X_1 *denote the time from 0 to the first event, and for* $n > 1$, *let* X_n *denote the time from the* $(n-1)st$ *to the* n*th event. Then* $X_n,\ n = 1, 2, \ldots$ *are independent identically distributed exponential random variables having mean* $\frac{1}{\lambda}$.

The above proposition which follows from the properties of stationary and independent increments implies that the process has no *memory* i.e., from any point in time onwards, the process is independent of all that has previously occurred (by independent increments) and also has the same distribution (by stationarity) as the original process. This memorylessness property of Poisson process is exploited quite often in stochastic analysis. We conclude our discussion with another useful property of the Poisson process. Suppose we are given that exactly one event has taken place in the interval $[0, t]$; then the time of the event has a *uniform* distribution over $[0, t]$. This can be generalized to more than one event (see [69, Theorem 2.3, page 17]).

2.3 Information Theory

Let $\{X_i\}$, $i \in \{0, 1, \ldots\}$ be a discrete (time) stochastic process. To define the entropy rate of $\{X_i\}$ we recall some basic information theory terms. The entropy $H(X)$ of a discrete random variable X with alphabet $\mathcal{H}$ and probability mass function $p(x) = \Pr\{X = x\}, x \in \mathcal{H}$ is

$$H(X) = -\sum_{x \in \mathcal{H}} p(x) \lg p(x) \tag{2.12}$$

The *joint entropy* $H(X_1, X_2)$ of a pair of discrete random variables (X_1, X_2) with a joint distribution $p(x_1, x_2)$ is

$$H(X,Y) = -\sum_{x_1 \in \mathcal{H}} \sum_{x_2 \in \mathcal{H}} p(x_1, x_2) \lg p(x_1, x_2) \tag{2.13}$$

The *conditional entropy* $H(X_2|X_1)$ is

$$\begin{aligned} H(X_2|X_1) = \sum_{x_1 \in \mathcal{H}} p(x_1) H(X_2|X_1 = x) = \\ -\sum_{x_1 \in \mathcal{H}} \sum_{x_2 \in \mathcal{H}} p(x_1, x_2) \lg p(x_2|x_1) \end{aligned} \tag{2.14}$$

The entropy per letter $H_n(\mathcal{H})$ of a stochastic process $\{X_i\}$ in a sequence of n letters is defined as

$$H_n(\mathcal{H}) = \frac{1}{n} H(X_1, X_2, \ldots, X_n) \tag{2.15}$$

Definition 2.3.1 *The entropy rate of a stochastic process* $\{X_i\}$ *is defined by*

$$H(\mathcal{H}) = \lim_{n \to \infty} H_n(\mathcal{H})$$

when the limit exists.

It can be shown that for *stationary processes* (with finite $H_1(\mathcal{H})$)) the limit $H(\mathcal{H})$ exists and

$$\begin{aligned} \lim_{n \to \infty} H_n(\mathcal{H}) = \\ \lim_{n \to \infty} H(X_n|X_{n-1}, X_{n-2}, \ldots, X_1) = H(\mathcal{H}) \end{aligned} \tag{2.16}$$

$$\begin{aligned} H(X_n|X_{n-1}, \ldots, X_1) \text{ is} \\ \text{non-increasing with } n \end{aligned} \tag{2.17}$$

$$H_n(\mathcal{H}) \geq H(X_n|X_{n-1}, \ldots, X_1) \tag{2.18}$$

$$H_n(\mathcal{H}) \text{ is non-increasing with } n \tag{2.19}$$

In particular when $X_1, X_2, \ldots$ are independent and identically distributed random variables (also called as a *discrete memoryless source*)

$$H(\mathcal{H}) = \lim \frac{1}{n} H(X_1, X_2, \ldots, X_n) = \lim \frac{nH(X_1)}{n} = H(X_1).$$

Informally, an *ergodic* source is one that cannot be separated into different persisting modes of behavior. A good illustration is given in [34]. More formally, the law of large numbers hold for ergodic processes. Let $\{X_n\}$ be an *ergodic stationary* process having a finite mean m. Then with probability one,

$$\lim_{n->\infty} \frac{1}{n}(X_1 + \cdots + X_n) = m \tag{2.20}$$

Good texts for an in-depth coverage of the above concepts are [23] and [34].

Chapter 3

Communication Networks

In this chapter, we study the problem of evaluating Quality-of-Service parameters in statistically multiplexed communication networks. Communication request is modeled by a simple stochastic source: an *on-off source*. Though simple, it captures the key feature of a *bursty* source, common in Internet traffic. Our main result is an efficient *Monte-Carlo* algorithm for estimating the failure probability of a general communication network. Our method is particularly useful in a dynamic setting in which communication requests are dynamically added and eliminated from the system. The amortized cost in our solution of updating the estimate after each change is proportional to the fraction of links involved in the change rather than to the total number of links in the network.

3.1 Introduction

Modern communication protocols such as ATM (Asynchronous Transfer Mode) achieve high utilization of channel bandwidth by multiplexing communication streams with different flow characteristics into one communication channel. Requests for communication are submitted to the network management protocol with some statistical characterization of the required communication. The network (flow) management protocol uses this information to *statistically* multiplex as many communication requests as possible while maintaining global network performance.

Next generation communication networks are expected to provide QoS (quality of service guarantees) when satisfying communication requests. In particular, QoS protocol is expected to limit to a pre-specified value the probability of communication failure due to events such as cell loss, cell delay and cell jitter [38]. The goal (on the part of the communication provider) is to satisfy as many communication requests as possible while maintaining pre-specified QoS guarantees. In this work, we focus on cell loss failures due to link or buffer overflow. Our method can be modified to handle other statistically governed communication characteristic such as cell delay and jitter.

In the case of a one link network, achieving QoS guarantees is reduced to bounding the probability that a sum of random variables exceeds a given bound, where each random variable represents the stream of data of one logical link, and the bound is the bandwidth capacity of the channel (or the

adjacent buffers). Most previous works have focused on communication flow modeled by an *on-off source* [49]. A (q, s) on-off source sends at a peak rate of s with probability q and zero otherwise. This model captures the extreme *bursty* nature of high speed networks based on ATM and related technologies. On-off sources have been studied extensively both in theory [49, 76, 50] as well as in simulation studies to evaluate performance of routing algorithms in ATM networks [38]. The work in [76] is especially interesting as it argues that the *fractal* nature of Internet and Ethernet traffic is captured very well by on-off sources. Techniques such as effective bandwidth give efficient and practical solution for providing QoS guarantees on one link networks [28].

The problem of bounding the overflow probability in a multi-channel network of arbitrary pattern is significantly harder. In fact the related optimization problem is $\#P$-complete [77] even if all logical links are fed by on-off sources with the same parameters q and s. We are not aware of any known heuristic to this problem, other than summing the failure probability on all network links. (Which is what is done in practice [38].) Such an estimate is far too expensive, since in large networks it can significantly over estimate overflow probability, and thus under utilizing network capacity.

In this work, we give the first non-trivial algorithms for estimating QoS properties in statistically multiplexed networks. Our algorithms employ efficient Monte-Carlo method to accurately estimate the overflow probability of any set of channels in an arbitrary topology network. These estimates translate into better admission control policy that achieve higher bandwidth utilization while guaranteeing same quality of service. We study this problem under *static* and *dynamic* settings. In the static setting, we assume that a set of connections are already active (routed) in the network and we are interested in whether QoS requirements are violated. In the dynamic setting, we are interested in admitting a new connection to the network and we desire a fast admission control algorithm to decide whether to admit this connection subject to QoS requirement.

Figure 3.1 illustrates the procedure followed by a typical network management protocol when admitting a new connection. When a new connection request, say C arrives, the routing algorithm (for example, an algorithm used in telephone networks is the Dynamic Routing Algorithm (DAR) [73]) selects a path in the network, say P where this connection might be routed. Then the admission control algorithm decides whether routing this connection through P would violate QoS guarantee. If QoS requirement is violated, then either C is rejected or (as in the DAR) an alternative route might be selected for P and the QoS guarantee condition is again checked. (In DAR the connection request is denied if there is a violation for the second time). If QoS guarantee is met, then connection request is admitted to the network. It is useful to think of admitting a connection in this way as it separates routing and admission control, although they might be implemented as a single algorithm [38]. In this work, we focus only on admission control. We assume that when the admission control protocol is invoked a path had already been selected by the routing algorithm.

3.1.1 ATM Networks and QoS Guarantees

To demonstrate the use of our estimation algorithms to achieve efficient admission control we focus in this section on its application to ATM networks.

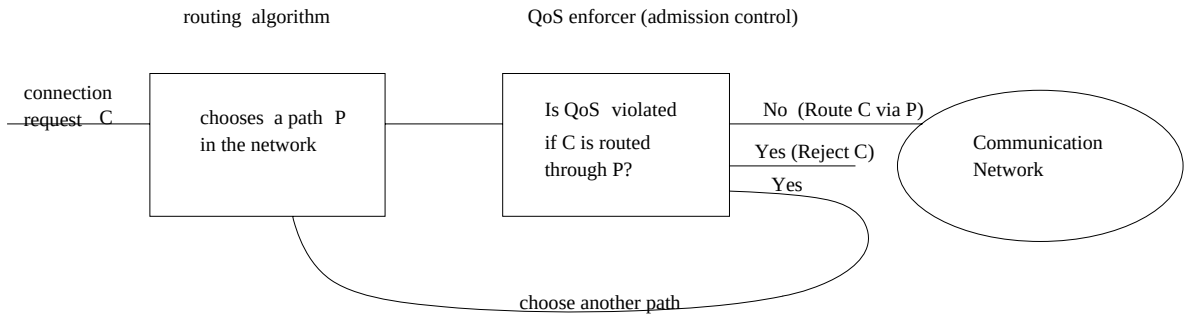

Figure 3.1: A schematic to illustrate how a connection is handled

ATM is emerging as a standard for Broadband Integrated Services Digital Networks (B-ISDN) carrying a wide spectrum of new consumer services such as video-on-demand, video teleconferencing etc. Specifically, ATM is a *high-speed, virtual-circuit-oriented packet-switching technique* that uses short, fixed-length packets called *cells*. A *virtual circuit* in an ATM network is a contract between the network and the customer to deliver traffic of specified *quality of service* (QoS).

Practical ATM networks are essentially modeled as a complete graph G of say, m edges and N nodes (terminals). Nodes are endpoints of a communication request and edges are communication links with specific bandwidth limits. A connection or a communication request is a (source, destination) pair with bandwidth requirements. Each edge in G can be thought of as a *Virtual Path* (VP) and connections are *Virtual Circuits* (VC's) ([38], [48]). When establishing a VC such that a pair of terminals can communicate, a route consisting of a set of VP's is selected. Following the practices of dynamic routing in telephone networks ([38], [48]), routes that consist of more than *two* hops are excluded. If a route consists of one VP, it is a *direct* route; otherwise, it is an *alternative route*.

To provide QoS guarantees, we suppose that the fraction of cells lost is not permitted to exceed a given fraction ρ.[1] Let $p_j(l)$ denote the fraction of cells lost in the input buffer of VP_j of the network, when l VCs are being routed through VP_j. We assume that these single link probabilities can be determined by cell level analysis [4] or by simulations [79]. We classify a VC as either a VC_j or a VC_{ij} if the VC is assigned one-edge route $\{j\}$ or two-edge route $\{i, j\}$, respectively. In an alternatively routed VC, cell loss can occur at either of the two VPs. Notice that

$$\Pr(\text{“cell lost on a } VC_{ij}\text{”} | l_i, l_j) \leq p_i(l_i) + p_j(l_j)$$

that is, the probability that a cell of a VC_{ij} is lost when there are l_i VCs using VP_i and l_j VCs using VP_j is less than the sum of two VP loss probabilities.

In [38], the *QoS-permissibility conditions* for setting up a new virtual circuit are as follows:

1. **direct route** j: (a) $p_j(l_j + 1) \leq \rho$ and
   (b) For every VP_k such that a VC_{jk} is in progress, $p_j(l_j + 1) + p_k(l_k) \leq \rho$.

[1] A typical value of ρ ranges from 10^{-6} to 10^{-9}.

Notice that the first of these conditions ensures that the additional VC will not cause cell loss to be excessive for any of the directly routed VCs on VP_j. The second condition ensures that cell loss will not be excessive for any of the "overlapping VCs" that is, the alternatively routed VCs employing VP_j.

2. **alternate route** $\{i, j\}$: (a) $p_i(l_i + 1) + p_j(l_j + 1) \leq \rho$ and
(b) For every VP_k such that a VC_{ik} is in progress, $p_i(l_i + 1) + p_k(l_k) \leq \rho$; and for every VP_k such that a VC_{jk} is in progress $p_j(l_j + 1) + p_k(l_k) \leq \rho$.

Notice that QoS-permissibility of any route involves examining VPs that are not on the route under consideration. In the above permissibility conditions in cases 1(b), 2(a) and 2(b) instead of just summing probabilities which can be a gross over estimation of the overall loss probability (especially when the two edges under consideration share a lot of common connections) we can use the methods developed here to accurately estimate this probability in these cases. Thus one of the main contributions of our work is to efficiently and accurately estimate such loss or overflow probabilities when we have arbitrary pattern of connections in an arbitrary topology network.

While we focus here on the problem of bounding the probability of an edge overflow in a given set of edges in the network (i.e. a subgraph of the network) our technique can be very easily adapted to address weaker QoS guarantees such as the *connection-based overflow constraint* [50]. A connection-based overflow constraint bounds the probability of an overflow along a given logical link. We can model overflow constraints in a network in two main ways: the *link-based overflow constraint* and the *connection-based overflow constraint* [50]. More formally, let the random variable X_i be the demand corresponding to connection c_i. Let P_i represent the *route* of the connection c_i, that is the edges of the path of c_i. The link-based overflow constraint requires that for each edge e_j, we have $\Pr[\sum_{i:e_j \in P_i} X_i > B_j] \leq Q_0$, where Q_0 is the overflow probability allowed by QoS. On the other hand, the connection-based overflow constraint requires that for each connection c_i we have $\Pr[\exists e_j \in P_i : \sum_{i:e_j \in P_i} X_i > B_j] \leq Q_0$. From the perspective of providing guaranteed quality of service to users in a network, the connection-based overflow constraint is more natural. Both these constraints can be enforced by our method of accurately estimating the overflow probability in an arbitrary subnetwork. For example, bounding the probability that an edge overflows in a suitable subgraph of the network enforces the connection-based overflow constraint. Under the QoS permissibility conditions mentioned earlier (which enforce the connection-based overflow constraint) we have to check for overflow condition in two-link subgraphs.

3.1.2 Problem Statement and Related Work

We consider a communication network $G = (V, E)$ with physical channels (edges) denoted by the set $E = \{e_1, e_2, \ldots, e_{|E|}\}$. For each channel e_i we have a constant B_i specifying the maximum *bandwidth* of that channel. The data flow in a *logical* channel, or a *communication request* or *connection* is characterized by a distribution function specifying the probability that the channel sends a given amount of data at a given time step. Thus a communication network consists of the physical network

G along with the connections active in the network. A *subnetwork* is simply a subset S of E along with the connections that are active in edges belonging to S. For our purposes, we simply model a communication request as a random variable. Our goal is to satisfy as many communication requests as possible while maintaining pre-specified *Quality of Service (QoS)* guarantees. We focus here on one such guarantee, bounding the probability of overflow on a given set of edges in the network (i.e. a given subnetwork).

Consider first a simple scenario in which all logical channels are fed by identical[2] (q, s) *on-off sources*, i.e. the data flow distribution in each logical channel has only two states: a flow of rate s with probability q, no flow with probability $1 - q$, and the states on different channels are independent events. Since all logical channels have identical, independent, distributions the probability of an overflow in a given edge can be computed by the Binomial distribution. Computing the probability of an overflow in the entire network for example, is significantly harder and can be shown to be $\#P$-complete by a straightforward reduction from the union of sets problem [46]. If the flow distribution of the logical channels is less restricted, then even the exact computation of overflow on one edge becomes intractable. In particular, if logical channels are fed by on-off sources with different parameters (q_i, s_i), even computing the overflow probability of one edge becomes $\#P$-complete [50].

In practice, the complexity of computing the overflow on one edge is circumvented by using the method of *effective bandwidth* [43, 49, 28]:

Definition 3.1.1 *The effective bandwidth of a random variable X is*

$$\beta_p(X) = \frac{\log E[p^{-X}]}{\log p^{-1}}. \tag{3.1}$$

The significance of the above definition will be clear from the following theorem which gives a conservative estimate of the overflow probability in the following sense: If the sum of the effective bandwidths of a set of independent connections does not exceed the link capacity, then the probability that the sum of their transmission rates exceeds twice the capacity at any instant is at most p. Given the effective bandwidth of individual logical channels a bound on the overflow probability of a given edge can be computed using the following result due to Hui[43]:

Theorem 3.1.1 *Let $X_1, \ldots, X_n$ be independent random variables, and $X = \sum_i X_i$. Let $a > b$. If $\sum_i \beta_p(X_i) \leq b$, then $\Pr\{X \geq a\} \leq p^{a-b}$.*

Proof: The proof is a simple application of Markov's inequality. From, $\sum_i \beta_p(X_i) \leq b$ we have $\sum_i \log E[p^{-X_i}] \leq \log p^{-b}$. Hence $\Pi_i E[p^{-X_i}] \leq p^{-b}$. Since X_i are independent, $\Pi_i E[p^{-X_i}] = E[\Pi_i p^{-X_i}] = E[p^{-\sum_i X_i}] = E[p^{-X}] \leq p^{-b}$. Now by Markov's inequality, $\Pr\{X \geq a\} = \Pr\{p^{-X} \geq p^{-a}\} \leq p^a E[p^{-X}] \leq p^{a-b}$. □

[2] Identical or almost identical connections capture ATM networks supporting *homogeneous sources* [38]. In ATM terminology this is referred to as a VP subnetwork which is a collection of VCs transporting identical sets of traffic sources i.e. with same traffic characteristics and QoS requirements. Statistical multiplexing is more effective for homogeneous traffic sources. See [38] for more details.

3.1.3 New Results

Given a method for estimating the overflow probability of a given edge (either directly for simple flow distributions, or through the concept of effective bandwidth for more general distributions) we are interested in estimating the overflow probability of a given set of edges in a network which is the probability that *some* edge will overflow among the set of edges (a more precise definition is given in section 3.2. More specifically we are interested in two versions of the problem:

1. **The Static Problem:** Given a set of edges S in a network and a set of logical channels estimate the overflow probability of the set S in the network.

2. **The Dynamic Problem:** Given a network (or a subnetwork), a set of logical channels, and a sequence of add and delete communication requests, determine for each request if granting that communication request violates a predetermined bound on network overflow probability in the network (or subnetwork).

Let m be the number of physical links in a subnetwork S, n be the number of active logical links (connections) in S (i.e. the connections that use any of the edges in S). Our results apply to any set of on-off sources (with possible different parameters to different logical links). For the static case we present an efficient (ϵ, δ)-approximation for overflow probability in S.

Definition 3.1.2 *An (ϵ, δ) Monte-Carlo approximation for Q is a value $\tilde{Q}$ such that*

$$\Pr[(1-\epsilon)Q \leq \tilde{Q} \leq (1+\epsilon)Q] \geq 1-\delta,$$

where the probability depends only on random steps made by the approximation algorithm.

Theorem 3.1.2 *There is a Monte-Carlo algorithm that computes an (ϵ, δ) approximation of the probability of overflow of a subnetwork S in $O(nm\epsilon^{-2}\log\delta^{-1})$ time.*

Henceforth, when we talk about an ϵ approximation *with high probability (whp)*, we mean that δ is $1/m^c$ for some constant $c \geq 1$. We refer to it as an ϵ approximation whp algorithm.

For the dynamic setting we define an *incremental* version of an ϵ approximation for determining QoS guarantee whp.

Definition 3.1.3 *Let Q_0 be the pre-defined QoS failure probability for a network (or subnetwork). Let Q' be the exact failure probability of the network (or subnetwork) after adding a new communication request. A dynamic algorithm is an ϵ approximation whp for QoS guarantee, if whp the algorithm rejects a new request when $Q' \geq (1-\epsilon)Q_0$, or accepts the new request when $Q' \leq (1+\epsilon)Q_0$.*

Theorem 3.1.3 *There is a Monte-Carlo ϵ approximation whp algorithm for the dynamic problem with $O(nf\log m)$ amortized complexity, where f is the number of edges involved in a given change in the communication (added or deleted requests).*

3.2 The Static Algorithm

Our static algorithm is based on the Karp, Luby and Madras (ϵ, δ) approximation algorithm for the cardinality of union of sets [46].

We will first briefly explain the algorithm for the union of sets problem. In the union of sets problem we are given m sets $D_1, ..., D_m$ and the goal is to estimate $|\cup_{i=1}^{m} D_i|$. The idea is to estimate the union by a Monte-Carlo sampling method as follows. Sample a pair (i, s), where $1 \leq i \leq m$, and $s \in D_i$, with probability $1/\sum_{i=1}^{m} |D_i|$ and compute the fraction of sets that contain s. By iterating this step $O(m)$ times one gets a tight estimate of the "overlap" between the sets, thus obtaining an estimate of the cardinality of the union. For the union of sets problem Karp and Luby prove the following theorem.

Theorem 3.2.1 *[46] There is a Monte-Carlo algorithm that computes an ϵ, δ approximation of the cardinality of the union of m sets in $(8 * (1 + \epsilon) * m \ln(3/\delta))/(1 - \epsilon^2/8)\epsilon^2$ steps.*

We will now describe the static algorithm. We will assume a communication network (graph) $G = (V, E)$ where all communications are fed by on-off sources with identical parameters, i.e. (q, s) on-off sources.

We will focus on a subnetwork $S \subseteq E$. [3] Let the number of edges in the subnetwork be m. In our formalization, instead of sets of elements we have m events, $\mathcal{E}_1, \ldots, \mathcal{E}_m$ where the event $\mathcal{E}_i$ is "edge $i \in E$ overflows". An event is a collection of states, where a *state* is defined as follows.

Definition 3.2.1 (state of the network) *Assume a network where all communications are fed by on-off sources. A state of the network is an on-off setting for all sources.*

For example a network fed by three on-off sources can be in state $s = (1, 0, 1)$, meaning that the first and third sources are in the "on" state, while the second state is in an "off" state. Given a state s, we denote its probability as $\Pr[s]$, which is simply the probability that the network will be in state s. Further, because we have assumed independence of sources, the above probability is the product of the probabilities of the individual sources being in their respective states. In the example above, $\Pr[s] = q^2(1 - q)$, assuming that all sources are identical and independent (q, s) on-off sources.

Thus the event $\mathcal{E}_i$ contains all the states (which we can think of as atomic events) that overflow edge i in S. Instead of estimating the cardinality of the union of sets we need to estimate the probability of the union of m events, where different states have different probabilities. That is we are interested in estimating Q, the subnetwork overflow probability which can be viewed as the probability that in a randomly chosen subnetwork state (according to the probability distribution of the connections) *some* edge in the subnetwork will overflow. We would like to calculate an (ϵ, δ) approximation of Q which we will denote by $\tilde{Q}$. (Henceforth, a tilde on top of a value denotes an estimate of that value in the (ϵ, δ) sense.)

[3] Of course S can be E itself, but to emphasize the full generality of the algorithm we describe the algorithm in terms of an arbitrary subnetwork. This also makes clear the fact that the algorithm does not depend on any way on the topology of the network or subnetwork, but only on the number of edges in it.

The static algorithm is given in figure 3.2. As is common in Monte-Carlo algorithms, the main idea is to set up a random variable (in our case Y_t) whose expectation is equal to the desired parameter (here Q) and then estimating the expectation by sampling from this random variable. For polynomial running time we have to show that we need only a polynomial number of samples to get an accurate estimate with high probability.

We now describe the algorithm informally and show how we set up the random variable Y_t. In theorem 3.2.2, we show formally that this random variable correctly estimates the required parameter Q. Let p_i be the probability that edge i overflows. This probability can be calculated exactly as we have assumed identical (q, s) on-off sources and the overflow probability in a single edge can be computed by the binomial distribution. Let $P = \sum_{i=1}^{m} p_i$. The algorithm consists of many *trial* steps which is iterated repeatedly till we get an (ϵ, δ) estimate of Q. One *trial* in the static algorithm (which is step 4 in figure 3.2) is choosing a pair (i, s), where $i \in E$ and $s \in \mathcal{E}_i$ (i.e. s is a state in which edge i overflows) with probability $\Pr[s]/P$, and estimating the fraction of number of edges that overflow in state s (this is similar to the "overlap" between sets mentioned in paragraph 2) In step 4.3 (consisting of 5 sub-steps), we compute the number of overflow edges in a fixed state s. If we do this step in a straightforward way, that is checking for overflow in each edge one by one, one trial itself will take $O(m)$ steps, as in the worst case we have to check every edge in S. Instead we resort to a Monte-Carlo sampling to estimate the number of overflow edges. This can be done simply by sampling edges uniformly at random and checking for overflow (step 4.3.4).

We will now mention how we choose a pair (i, s) at the beginning of the trial step. This is done in two steps. In step 4.1 we first choose an edge $i \in E$ with probability p_i/P and then choose an *overflow* state $s \in \mathcal{E}_i$ with probability $Pr[s]/p_i$ in step 4.2. We use the *choose* algorithm in selecting a random overflow state $s \in \mathcal{E}_i$ with the appropriate probability i.e. $\Pr[s]/p_i$. Let $C_i = \{c_1, \ldots, c_{|C_i|}\}$ be the set of connections going through edge i. Given an edge i we choose a state that overflows that edge using the algorithm *choose* given below.

(Let $\bar{\mathcal{E}}$ denote the complement of the event $\mathcal{E}$)
Algorithm *choose*: choosing a state s with probability $\frac{\Pr[s]}{p_i}$

1 Set all connections not belonging to edge C_i to "on" with probability q.
and "off" with probability $1 - q$
2 Let $\mathcal{F}_0$ be the event "edge i overflows"
3 **for** $k = 1$ **to** $|C_i|$ **do**
3.1 Let $\mathcal{E}_k$ be the event "c_k is on"
3.2 Set c_k to "on" with probability $b_k = \Pr\{\mathcal{E}_k | \mathcal{F}_{k-1}\}$
and to "off" with probability $1 - b_k$.
3.3 **if** c_k is set to "off" **then** $\mathcal{F}_k = \bar{\mathcal{E}}_k \cap \mathcal{F}_{k-1}$
3.4 **else** $\mathcal{F}_k = \mathcal{E}_k \cap \mathcal{F}_{k-1}$

Lemma 3.2.1 *The choose algorithm chooses an overflow state s with the probability* $\Pr[s]/p_i$.

Proof: Connections outside C_i are clearly independent of the event "edge i overflows". Hence these connections can be set to "on" with probability q and to "off" with probability $1-q$. Connections belonging to C_i are set to "on" or "off" with probabilities as calculated in the choose algorithm. Suppose s is a state in which edge i overflows. Let $s(c_i)$ be the status ("on" or "off") of connection c_i in state s and let $\Pr[\mathcal{E}_{s(c_i)}]$ denote the probability of the event that c_i is in state $s(c_i)$. Then the probability that a connection c_i is set to $s(c_i)$ is $\Pr[\mathcal{E}_{s(c_i)}|F_0 \cap \mathcal{E}_{s(c_1)} \cap \cdots \cap \mathcal{E}_{s(c_{i-1})}]$. By the rule of conditional probability we have,

$$\Pr[\mathcal{F}_0] \times \Pr[\mathcal{E}_{s(c_1)}|\mathcal{F}_0] \times \Pr[\mathcal{E}_{s(c_2)}|\mathcal{F}_0 \cap \mathcal{E}_{s(c_1)}] \times \cdots \times \Pr[\mathcal{E}_{s(c_{|C_i|})}|\mathcal{F}_0 \cap \mathcal{E}_{s(c_1)} \cap \cdots \cap \mathcal{E}_{s(c_{n-1})}] = \Pr[\mathcal{F}_0 \cap \mathcal{E}_{s(c_1)} \cap \cdots \cap \mathcal{E}_{s(c_{|C_i|})}] = \Pr[s]$$

since in state s the edge overflows. Thus probability that state s is chosen is $\frac{Pr[s]}{p_i}$ since $\Pr[\mathcal{F}_0] = p_i$. □

The conditional probabilities in the choose algorithm can be calculated using the Bayes' rule as follows. For example the probability that a connection c is "on" in a randomly chosen overflow state of a particular edge i is given by Bayes' rule as:

$$\Pr\{c \text{ is } on | i \text{ } overflows\} = \frac{\Pr\{i \text{ } overflows | c \text{ is "}on''\} * \Pr\{c \text{ is "}on''\}}{\Pr\{i \text{ } overflows\}}$$

The choose algorithm can be implemented in $O(n)$ time by pre-computing the conditional probabilities and using a look up table.

In the following theorem we show that $E[Y_t]$ is Q. Also the running time is upper bounded by N which is polynomial in $m, 1/\epsilon$ and $\log(1/\delta)$.

Theorem 3.2.2 *The run-time of the static algorithm given in Figure 1 is $O(nm\epsilon^{-2}\log\delta^{-1})$ and it produces an (ϵ,δ) approximation of the overflow probability in the subnetwork S.*

Proof: We show that $E[Y_t] = Q$ for any index t. Let s be an overflow state in S. Let $C(s) = \{(s,i) : \text{edge } i \text{ overflows in state } s\}$. Let $R_k = \{s : |C(s)| = k\}, k = 1, \ldots, m$. That is R_k is the set of all overflow states in which exactly k edges overflow. Let $r_k = \sum_{s \in R_k} \Pr(s)$. Then, $P = \sum_{k=1}^{m} k * r_k$ and $Q = \sum_{k=1}^{m} r_k$. This is because $P = \sum_{i=1}^{m} p_i$ is the sum of the probabilities of all overflow states s with s being counted k times if $s \in R_k$ (once for each edge). On the other hand Q is simply the sum of probabilities of all overflow states s. Now, $E[Y_t] = E[t(s)] * P/m$. Since the random variable $t(s)$ depends only on k, $t(s)$ is geometrically distributed with mean m/k. Hence, $E[Y_t] = \sum_{k=1}^{m} \Pr(s \in R_k) * P/k$. Steps 4.1 and 4.2 choose a state s with probability $Pr[s]/P$. Hence, the probability that in a trial a state $s \in R_k$ is chosen is $k * r_k/P$, and we have $E[Y_t] = Q$. The random variables $Y_1, \ldots, Y_t$ are independent and identically distributed with mean Q. We obtain an estimate of the mean by simply averaging over these t random variables. The number of steps needed to get an (ϵ,δ) approximation follows due to theorem 3.2.1 proved in [46]. □

The Static Algorithm
(m is the size of the subnetwork)

1 $gtime = 0$ /* $gtime$ counts the global number of steps executed */
2 $t = 0$ /* counts the number of trials */
3 $N = (8 * (1 + \epsilon) * m \ln(3/\delta))/(1 - \epsilon^2/8)\epsilon^2$
4 **trial:**
    4.1 randomly choose $i \in \{1, \ldots, m\}$ such that i is chosen with probability p_i/P
    4.2 choose a state s with probability $Pr[s]/p_i$ using the choose algorithm
    4.3 $t(s) = 0$ /* $t(s)$ is the *step number*, counts the number of steps in this trial */
    step:
        4.3.1 $t(s) = t(s) + 1$
        4.3.2 $gtime = gtime + 1$
        4.3.3 **if** $gtime > N$ **then go to** 5
        4.3.4 randomly choose $j \in \{1, \ldots, m\}$ with probability $1/m$
        4.3.5 **if** j does not overflow in s **then go to step**
    4.5 $t = t + 1$
    4.6 $Y_t = P * t(s)/m$
    4.7 **go to trial**
5 $\tilde{Q} = \frac{\sum_{j=1}^{t} Y_t}{t}$

Figure 3.2: Static Algorithm for estimating overflow probability in a subnetwork

We can use the static algorithm even when connections are arbitrary on-off random variables by using Hui's theorem (theorem 3.1.1) as an upper bound on the overflow probability. For example, we can choose b in the above theorem to be the bandwidth capacity of the edge and a to be $2b$. Then Hui's theorem gives an upper bound on the probability that twice the bandwidth capacity will be exceeded. Then our Monte-Carlo algorithm will find an overall estimate of this upper bound. To choose a state s with the appropriate probability in the choose algorithm we again use Hui's theorem when calculating the conditional probabilities.

3.3 The Dynamic Algorithm

We consider now the dynamic problem in which the network protocol needs to react to a sequence of add and delete requests. For each add request, the protocol needs to decide if adding that request violates the system's QoS requirement. The goal is to use past estimates in order to minimize the work of each evaluation. Assume that the change in the network (add or delete) involves a subset $F \subseteq E$ of edges, where E is the total number of edges in the network. In this section for simplicity, we focus on estimating the overflow probability of the entire network. The same analysis carries over if we focus on the overflow probability of any subnetwork. Since the static algorithm requires $O(nm \log m)$ steps [4] to check for QoS guarantee without prior information, we are looking for an

[4] From now on, we assume that δ is $1/m^c$ for some $c \geq 1$ to guarantee estimation with high probability. ϵ is a fixed constant. Also $|E|$, the size of the network is assumed to be m.

incremental algorithm that can check the same in $|F|/m$ of that complexity or $O(n|F|\log m)$ steps. Since error probabilities accumulate, we will need to compute a new estimate for the whole network after a long sequence of changes, however, the amortized complexity will remain $O(n|F|\log m)$ work per addition or deletion.

Let Q and Q' denote the overflow probability of the network before and after the connection was added. Let $\tilde{Q}$ and $\tilde{Q}'$ denote their respective estimates. Let Q_F and Q'_F denote the probability that an edge in F overflows in a randomly chosen state before and after the change. We also define Q_{E-F} to be probability that *only* an edge in $E-F$ overflows in a randomly chosen network state. Clearly

$$Q = Q_F + Q_{E-F} \tag{3.2}$$

And since the change involves only edges in F

$$Q' = Q'_F + Q_{E-F} \tag{3.3}$$

Let W be the probability that a random state s that overflows before the change, overflows *only* edges in $E-F$, then $Q_{E-F} = WQ$. Thus, instead of estimating Q' directly (which requires $O(nm\log m)$ time) we can estimate Q'_F and W, focusing only on $|F|$ edges. The *incremental algorithm* for estimating the overflow probability after a change that involves the set of edges F is given in Figure 3.3.

The main result of this section is the following theorem.

Theorem 3.3.1 *The incremental algorithm of Figure 2 satisfies definition 3.1.3 and takes $O(n|F|\log m)$ steps, where n is the total number of connections in the network.*

Proof: We will show that the algorithm gives an ϵ approximation of Q'. The idea used is as follows. Since we use the formula $\tilde{Q}' = \tilde{Q}'_F + \tilde{W} * \tilde{Q}$, we lose some accuracy because the second term is in the form of a product and we have assumed that we have an ϵ estimate of Q. To compensate for this loss, we estimate Q'_F with a greater accuracy, which can be done if Q'_F is not "too small". On the other hand, if Q'_F is "very small" we can directly bound Q'.

More precisely, we have an ϵ approximation of Q and ϵ' approximations for Q'_F and W. This gives rise to eight different cases. We will analyze the most interesting among them. We will show that

$$(1-\epsilon)Q' \leq \tilde{Q}' = (1+\epsilon')Q'_F + (1+\epsilon)Q(1+\epsilon')W \leq (1+\epsilon)Q' \tag{3.4}$$

The left side of the inequality is obvious. We will show the right side. That is we have to show that,

$$\epsilon' * Q'_F + Q * W(\epsilon + \epsilon' + \epsilon\epsilon') \leq \epsilon * Q' \tag{3.5}$$

Simplifying we have,

$$\epsilon' \leq \frac{\epsilon * Q' - \epsilon * Q * W}{Q'_F + (1+\epsilon)WQ} = \frac{\epsilon Q'_F}{Q'_F + (1+\epsilon)WQ} \tag{3.6}$$

Incremental Algorithm
(Q_0 is a parameter fixed by QoS. $\tilde{Q}$ is the current ϵ estimate of the network overflow probability. $\tilde{Q}$ is the new estimate after a connection is added or deleted.
Fix $\epsilon' = \frac{\epsilon^2}{2(1+\epsilon)}$ and $\delta' = 1/(3m^3)$. $FLAG$ is a global logical variable)

1 Estimate the overflow probability $\tilde{Q'}_F$ for the set F (after the new connection was added) using the algorithm of Figure 1, with the parameters (ϵ', δ').
If $\frac{\tilde{Q'}_F}{1-\epsilon'} \leq \frac{\epsilon \tilde{Q}}{2(1+\epsilon)}$ **then**
  $\tilde{Q}' = \tilde{Q}$
  set $FLAG$ = "true"
  exit
else set $FLAG$ = "false" and go to step 2.
2 Let W be the probability that a randomly chosen network state overflows (before the new connection was added or deleted) *only* in $E - F$ and not in F.
The following steps compute an (ϵ', δ') approximation $\tilde{W}$ for W:
2.1 **for** $k = 1$ **to** $N = \frac{2+\epsilon}{\epsilon-\epsilon'} * (4\ln(2/\delta')/(\epsilon')^2)$ **do**
    2.1.1 randomly choose $i \in \{1, \ldots, m\}$ such that i is chosen with probability $p_i/\tilde{Q}$
    2.1.2 choose a state s with probability $Pr[s]/p_i$
    (The above two steps choose an "overflow" state s with probability $Pr[s]/\tilde{Q}$)
    2.1.3 **if** any edge in F overflows **then** set $Z_k = 0$
    2.1.4 **else** set $Z_k = 1$
2.2 **endfor**
2.3 $\tilde{W} = \sum_k Z_k/N$
3 $\tilde{Q}' = \tilde{Q'}_F + \tilde{W} * \tilde{Q}$
4 **If** $\tilde{Q}' \leq Q_0$ **then**
    "QoS is not violated: admit connection request"
else
    "QoS is violated: reject connection request"

Figure 3.3: Incremental Algorithm for determining QoS guarantee after a connection is added or deleted

Now, by our condition in step 1, whp we have $Q'_F/Q \geq \epsilon/2$. Using the fact that $W = 1 - Q_F/Q$, we can show that the above inequality holds.

Now we can show that this gives an incremental ϵ approximation for determining QoS guarantee. This is clear if we proceed to step 2, since we compute Q' with ϵ accuracy. On the other hand, if we finish at step 1, then because of the condition stated, $Q'_F \leq \frac{\epsilon Q}{2}$ whp. Since $Q' \leq Q + Q'_F$, we still don't violate the QoS guarantee whp.

Finally, we show that running time of the algorithm is $O(n|F|\log m)$. Step 1 takes $O(n|F|\log m)$ time. Step 2 takes $O(n|F|\log m)$ time because we have to check for overflow only in edges belonging to F (in step 2.1.3). The upper bound we choose for the number of trials needed to estimate W needs some explanation. Step 2 is simply an application of the zero-one estimator theorem of [46]. To bound the number of trials needed, we calculate a lower bound on W needed for an ϵ estimation of Q'. We do this as follows. We again have eight different cases, but the inequality in 3.4 gives the

lower bound on W. Strengthening inequality 3.5 we require,

$$\epsilon' * Q' + Q' * W(\epsilon + \epsilon' + \epsilon\epsilon') \leq \epsilon * Q' \tag{3.7}$$

because $Q'_F \leq Q'$ and $Q \leq Q'$. So we have,

$$W \leq \frac{\epsilon - \epsilon'}{2 + \epsilon} \tag{3.8}$$

Hence it's enough if we perform $N = \frac{2+\epsilon}{\epsilon-\epsilon'} * (4\ln(2/\delta')/(\epsilon')^2)$ trials. □

We notice that in the incremental algorithm we "lose" a small constant factor in the confidence probability in each call to the algorithm. This is because our estimate in the incremental algorithm (step 3) is an addition of two terms, one of which is in the form of a product. Hence if the error in each estimate is δ, the error of the total estimate adds up to $3*\delta$. So, to maintain an ϵ approximation whp, we run the first and all calls of the incremental algorithm with $\delta = 1/(3m^3)$. Every m^2 calls we re-evaluate Q using the static algorithm for the entire network (with $\delta = 1/(3m^3)$). Since a call to the static algorithm takes $O(mn\log m)$ time the amortized work done for every addition or deletion is still $O(n|F|\log m)$.

We use the logical variable $FLAG$ to check whether we proceeded to step 2 or not in the incremental algorithm. In other words, if $FLAG$ = "true", then we have $Q'_F/Q \leq \epsilon/2$ with high probability, and we didn't proceed to step 2. Although this means, that we are still within the "safe" range as far as QoS guarantee is concerned, we have to keep track of additions or deletions, where this occurs. Otherwise, errors can accumulate. However, note that once we proceed to step 2 of the incremental algorithm ($FLAG$ will then be "false") we will have ϵ accurate estimate of Q'. Hence till this happens, F will be *union* of edges of all new connections added/deleted since the last time $FLAG$ was "false". Again, this does not change the amortized complexity of $O(n|F|\log m)$ work done per addition or deletion.

Chapter 4

Peer-to-Peer Networks

In this chapter we design a practical and scalable distributed protocol for building connected low-diameter Peer-to-Peer networks. We analyze the protocol under a realistic dynamic setting where the arrival of peers is according to a Poisson process and the duration of time a peer stays connected to the network is independently and exponentially distributed. Our stochastic analysis shows that our protocol results in *constant* degree networks that are very likely to stay *connected* with diameter *logarithmic* in the size of the network. The technical contribution of our analysis is a non-trivial stochastic analysis of an evolving graph with nodes and edges arriving and leaving the network.

4.1 Introduction

Peer-to-peer (or "P2P") networks are emerging as a significant vehicle for providing distributed services (e.g., search, content integration and administration) both on the Internet [20, 21, 22, 40] and in enterprises. The idea is simple: rather than have a centralized service (say, for search), each node in a distributed network maintains its own index and search service. Queries no longer go to a central server; instead they fan out over the network, and results are collected and propagated back to the originating node. This allows for search results that are fresh (in the extreme, admitting dynamic content assembled from a transaction database, reflecting – say in a marketplace – real-time pricing and inventory information). Such freshness is not possible with traditional static indices, where the indexed content is as old as the last crawl (in many enterprises, this can be several weeks). The downside, of course, is dramatically increased network traffic. In some implementations [21] this problem can be mitigated by adaptive distributed caching for replicating content; it seems inevitable that such caching will become more widespread.

How should the topology of P2P networks be constructed? Unlike static networks, P2P systems are very dynamic with a high peer turnover rate [71]. For example in both Gnutella [39] and Napster [57], about half of the peers participating in the system are replaced within one hour. Thus maintaining even basic property such as network connectivity is a non-trivial task.

Each node participating in a P2P network runs so-called *servent* software (for *ser*ver+cli*ent*, since

every node is both a server and a client). This software embeds local heuristics by which the node decides, on joining the network, which neighbors to connect to. Note that an incoming node (or for that matter, any node in the network) does not have global knowledge of the current topology, or even the identities (IP addresses) of other nodes in the current network. Thus one cannot require an incoming node to connect (say) to "four random network nodes" (in the hope of creating an expander-like network). What local heuristics will lead to the formation of networks that perform well? Indeed, what properties should the network have in order for performance to be good? In the Gnutella world [40] there is little consensus on this topic, as the variety of servent implementations (each with its own peculiar connection heuristics) grows – along with little understanding of the evolution of the network. Indeed, some services on the Internet [19] attempt to bring order to this chaotic evolution of P2P networks, but without necessarily using rigorous approaches (or tangible success).

A number of attempts are under way to create P2P networks within enterprises (e.g., Verity is creating a P2P enterprise infrastructure for search). The principal advantage here is that servents can be implemented to a standard, so that their local behavior results in good global properties for the P2P network they create. In this work we begin with some desiderata for such good global properties, principally the diameter of the resulting network (the motivation for this becomes clear below). Our main contribution is a stochastic analysis of a simple local heuristic which, if followed by every servent, results in provably strong guarantees on network diameter and other properties. Our heuristic is intuitive and practical enough that it could be used in enterprise P2P products.

4.1.1 Case study: Gnutella

To better understand the setting, modeling and objectives for the stochastic analysis to follow, we now give an overview of the Gnutella network. This is a public P2P network on the Internet, by which anyone can share, search for and retrieve files and content. A participant first downloads one of the available (free) implementations of the search servent. The participant may choose to make some documents available for public sharing, and indexes the contents of these documents and runs a search server on the index. His servent joins the network by connecting to a small number (typically 3-5) of neighbors currently connected to the network. When any servent s wishes to search the network with some query q, it sends q to its neighbors. These neighbors return any of their own documents that match the query; they also propagate q to *their* neighbors, and so on. To control network traffic this fanning-out typically continues to some fixed radius (in Gnutella, typically 7); matching results are fanned back into s along the paths on which q flowed outwards. Thus every node can initiate, propagate and serve query results; clearly it is important that the content being searched for be within the search radius of s. A servent typically stays connected for some time, then drops out of the network – many participating machines are personal computers on dialup connections. The importance of maintaining connectivity and small network diameter has been demonstrated in a recent performance study of the public Gnutella network [19].

Note that the above discussion lacks any mention of which 3-5 neighbors a servent joining the

network should connect to; and indeed, this is the current free-for-all situation in which each servent implementation uses its own heuristic. Most begin by connecting to a generic set of neighbors that come with the download, then switch (in subsequent sessions) to a subset of the nodes whose names the servent encountered on a previous session (in the course of remaining connected and propagating queries, a servent gets to "watch" the names of other hosts that may be connected and initiating or servicing queries). Note also that there is no standard on what a node should do if its neighbors drop out of the network (many nodes join through dialup connections, and typically dial out after a few minutes – so the set of participants keeps changing). This free-for-all situation leads to the fragmentation of the network into disconnected pieces documented in [19].

4.1.2 Main Contributions and Organization of the Chapter

Our main contribution is a new protocol by which newly arriving servents decide which network nodes to connect to, and existing servents decide when and how to replace lost connections. We show that our protocol results in a constant degree network that is likely to stay connected and have small diameter. A nice feature of our protocol is that it operates without any global knowledge (such as the topology of the network or even the identities of all other nodes) and can be implemented by a simple distributed local message passing scheme. Also our protocol is easily scalable both in terms of degree (which remains bounded irrespective of size) and diameter (grows slowly as a function of network size).

Our protocol for building a P2P network is described in Section 4.2. Sections 4.3 presents a stochastic analysis of our protocol. Our protocol involves one somewhat non-intuitive notion, by which nodes maintain "preferred connections" to other nodes; in Section 4.4 we show that this feature is essential. Our analysis assumes a stochastic setting in which nodes arrive and leave the network according to a probabilistic model. Our goal is to show that even as the network changes with these arrivals/departures, it remains connected with small diameter. Our main result is that at *any* time (after a short initial period), with large probability, the network is *connected* and its diameter is *logarithmic* in the size of the network at that time. Furthermore, our analysis proves that the protocol has strong fault tolerance properties, if the network is fragmented into disconnected pieces it rapidly recovers its connectivity. The technical core of our analysis is an analysis of an evolving graph as nodes arrive and leave, with edges being dictated by the protocol; the analysis of evolving graphs is relatively new, with virtually no prior analysis in which both nodes and edges arrive and leave the network.

We mention related work in Section 4.5 and discuss open issues in Section 4.6.

4.2 The P2P Protocol

The central element of our protocol is a *host server* [1] which, at all times, maintains a *cache* of K nodes, where K is a constant. The host server is reachable by all nodes at all times; however, it need not know of the topology of the network at any time, or even the identities of all nodes currently on the network. We only require that (1) when the host server is contacted on its IP address it responds, and (2) any node on the P2P network can send messages to its neighbors. In this sense, our protocol demands far less from the network than do (for instance) current P2P proposals (e.g., the *reflectors* of dss.clip2.com, which maintain knowledge of the global topology).

When a node is in the cache we refer to it as a *cache node*. A node is *new* when it joins the network, otherwise is is *old*. Our protocol will ensure that the degree (number of neighbors) of all nodes will be in the interval $[D, C+1]$, for two constants D and C.

A new node first contacts the host server, which gives it D random nodes from the current cache to connect to. The new node connects to these, and becomes a *d-node*; it remains a d-node until it subsequently either enters the cache or leaves the network. The degree of a d-node is always D. At some point the protocol may put a d-node into the cache. It stays in the cache until it acquires a total of C connections, at which point it leaves the cache, as a *c-node*. A c-node might lose connections after it leaves the cache, but its degree is always at least D. A c-node has always one *preferred* connection, made precise below. Our protocol is summarized below as a set of rules applicable to various situations that a node may find itself in.

Peer-to-Peer Protocol for Node v:

1. **On joining the network:** Connect to D cache nodes, chosen uniformly at random from the current cache.

2. **Reconnect rule:** If a neighbor of v leaves the network, and that connection was not a preferred connection, connect to a random node in cache with probability $D/d(v)$, where $d(v)$ is the degree of v before losing the neighbor.

3. **Cache Replacement rule:** When a cache node v reaches degree C while in the cache (or if v drops out of the network), it is replaced in the cache by a d-node from the network. Let $r_0(v) = v$, and let $r_k(v)$ be the node replaced by $r_{k-1}(v)$ in the cache. The replacement d-node is found by the following rule:

   $k = 0$;
   while (a d-node is not found) **do**
   search neighbors of $r_k(v)$ for a d-node;
   $k = k + 1$;
   endwhile

[1] The host server is similar to (or models) websites that maintain list of host IP addresses which clients visit to get entry points into the P2P network; for example, *http://www.gnufrog.com/* is a website which maintains a list of active Gnutella servents. New client can join the network by connecting to a one or more of these servents.

4. **Preferred Node rule:** When v leaves the cache as a c-node it maintains a *preferred connection* to the d-node that replaced it in the cache. (If v is not already connected to that node this adds another connection to v.)

5. **Preferred Reconnect rule:** If v is a c-node and its preferred connection is lost, then v reconnects to a random node in the cache and this becomes its new preferred connection.

We end this section with brief remarks on the protocol and its implementation.

1. It is clear from our protocol that it is essential for a node to know whether it is in the cache or not; thus each node maintains a flag for this purpose.

2. The cache replacement rule can be implemented in a distributed fashion by a local message passing scheme with constant storage per node. Each c-node v stores the address of the node that replaced it in the cache, i.e., $r(v)$. Node v sends a message to $r(v)$ when v itself doesn't have any d-node neighbors.

3. Note that the overhead in implementing each rule of the protocol is constant (or expected constant). This is very important in practice, because even if a protocol is local, it is desirable that neither too much (local) computation nor too many local messages be sent per node. Rules 1, 2, 4 and 5 can be easily implemented with constant overhead. It follows from our analysis that the overhead incurred in replacing a full cache node (rule 3) is constant on the average, and with high probability is at most logarithmic in the size of the network (see Section 4.3.2).

4. We note that the host server is contacted whenever a node needs to reconnect (rules 2 and 5), and when a new node joins the network. We show that the expected number of contacts the host server receives per unit time interval is constant in our model and with high probability only *logarithmic* in the size of the network; this implies that the network also scales well in terms of the number of "hits" the central server receives.

5. In the stochastic analysis that follows, the protocol does have a minuscule probability of catastrophic failure: for instance, in the cache replacement step, there is a very small probability that no replacement d-node is found. A practical implementation of this step would either cause some nodes to exceed the maximum capacity of $C + 1$ connections, or to reject new connections. In either case, the system would rapidly "self-correct" itself out of this situation (failing to do so with an even more minuscule probability). For either such implementation choice, our analysis can be extended.

4.3 Analysis

In evaluating the performance of our protocol we focus on the long term behavior of the system in a fully decentralized environment in which nodes arrive and depart in an uncoordinated, and

unpredictable fashion. This setting is best modeled by a stochastic, memoryless, continuous-time setting. The arrival of new nodes is modeled by Poisson distribution with rate λ, and the duration of time a node stays connected to the network is independently and exponentially distributed with parameter μ. We are inspired by models in queuing theory which have been used to model similar scenarios, e.g., the classical telephone trunking model [45]. A recent measurement study of real P2P systems [71] (– Gnutella and Napster) provides evidence that our model approximates real-life data fairly well.

Let G_t be the network at time t (G_0 has no vertices). We analyze the evolution in time of the stochastic process $\mathcal{G} = (G_t)_{t \geq 0}$.

Since the evolution of $\mathcal{G}$ depends only on the ratio λ/μ we can assume w.l.o.g. that $\lambda = 1$. To demonstrate the relation between these parameters and the network size, we use $N = \lambda/\mu$ throughout the analysis. We justify this notation in the next section by showing that the number of nodes in the network rapidly converges to N. Furthermore, if the ratio between arrival and departure rates is changed later to $N' = \lambda'/\mu'$, the network size will then rapidly converge to the new value N'. Next we show that the protocol can w.h.p.[2] maintain a bounded number of neighbors for all nodes in the network, i.e., w.h.p. there is a d-node in the network to replace a cache node that reaches full capacity. In Section 4.3.3 we analyze the connectivity of the network, and in Section 4.3.4 we bound the network diameter.

4.3.1 Network Size

Let $G_t = (V_t, E_t)$ be the network at time t.

Theorem 4.3.1 *1. For any $t = \Omega(N)$, w.h.p. $|V_t| = \Theta(N)$.*

2. If $\frac{t}{N} \to \infty$ then w.h.p. $|V_t| = N + o(N)$.

Proof: Consider a node that arrived at time $\tau \leq t$. The probability that the node is still in the network at time t is $e^{-(t-\tau)/N}$. Let $p(t)$ be the probability that a random node that arrives during the interval $[0, t]$ is still in the network at time t, then (since in a Poisson process the arrival time of a random element is uniform in $[0, t]$),

$$p(t) = \frac{1}{t} \int_0^t e^{-(t-\tau)/N} d\tau = \frac{1}{t} N(1 - e^{-t/N}).$$

Our process is similar to an infinite server Poisson queue. Thus, the number of nodes in the graph at time t has a Poisson distribution with expectation $tp(t)$ (see [69],pages 18–19).

For $t = \Omega(N)$, $E[|V_t|] = \Theta(N)$. When $t/N \to \infty$, $E[|V_t|] = N + o(N)$.

We can now use a tail bound for the Poisson distribution (2.8) to show that for $t = \Omega(N)$,

$$Pr\left(||V_t| - E[|V_t|]| \leq \sqrt{bN \log N}\right) \geq 1 - 1/N^c$$

[2] Throughout this extended abstract w.h.p. denotes probability $1 - N^{-\Omega(1)}$.

for some $c > 1$. □

The above theorem assumed that the ratio $N = \lambda/\mu$ was fixed during the interval $[0, t]$. We can derive similar result for the case in which the ratio changes to $N' = \lambda'/\mu'$ at time τ.

Theorem 4.3.2 *Suppose that the ratio between arrival and departure rates in the network changed at time τ from N to N'. Suppose that there were M nodes in the network at time τ, then if $\frac{t-\tau}{N'} \to \infty$ w.h.p. G_t has $N' + o(N')$ nodes.*

Proof: The expected number of nodes in the network at time t is

$$Me^{-\frac{(t-\tau)}{N'}} + N'(1 - e^{-\frac{t-\tau}{N'}}) = N' + (M - N')e^{-\frac{t-\tau}{N'}}.$$

Applying the tail bound for the Poisson distribution we prove that w.h.p. the number of nodes in G_t is $N' + o(M - N')$. □

4.3.2 Available Node Capacity

To show that the network can maintain a bounded number of connections at each node we will show that w.h.p there is always a d-node in the network to replace a cache node that reaches capacity C, and that the replacement node can be found efficiently. We first show that at any given time the network has w.h.p. a large number of d-nodes.

Lemma 4.3.1 *Let $C > 3D$; then at any time $t \geq a \log N$, (for some fixed constant $a > 0$), w.h.p. there are*

$$(1 - \frac{2D}{C - D}) \min[t, N]$$

d-nodes in the network.

Proof: Assume that $t \geq N$ (the proof for $t < N$ is similar). Consider the interval $[t - N, t]$; we bound the number of new d-nodes arriving during this interval and the number of nodes that become c-nodes.

The arrival of new nodes to the network is Poisson-distributed with rate 1; using the tail bound for the Poisson distribution we show that w.h.p the number of new d-nodes arriving during this interval is $N(1 + o(1))$, and that the number of connections to cache nodes from the new arrivals is $DN(1 + o(1))$.

By Theorem 4.3.1 w.h.p. there were never more than $N(1 + o(1))$ nodes in the network at any time in this interval. Thus, the number of nodes leaving the network in this interval is Poisson-distributed with expectation $\leq N(1 + o(1))$ and w.h.p. no more than $N(1 + o(1))$ nodes left the network in the interval. The expected number of connections to the cache from old nodes is bounded by

$$N(1 + o(1)) \sum_{v \in V} \frac{d(v)}{N} \frac{D}{d(v)} = ND(1 + o(1)).$$

Let $u_1, ..., u_\ell$ be the set of nodes that left the network in that interval, and let $X_{v,u_i} = 1$ if v makes connection to the cache when u_i left the network, else $X_{v,u_i} = 0$. Then

$$E\left[\sum_{j=1}^{\ell}\sum_{v} X_{v,u_i}\right] = ND(1+o(1))$$

and each variable in the sum is independent of all but C other variables. By partitioning the sum into C sums such that in each sum all variables are independent, and applying the Chernoff bound (2.6) to each sum individually, we show that w.h.p. the total number of connections to the cache from old nodes during this interval is bounded w.h.p by $ND(1+o(1))$.

Since a node receives $C-D$ connections while in the cache, w.h.p. no more than $\frac{2D}{C-D}N$ d-nodes convert to new c-nodes in the interval; thus w.h.p we are left with $(1-\frac{2D}{C-D})N$ d-nodes that joined the network in this interval. □

Lemma 4.3.2 *Suppose that the cache is occupied at time t by node v. Let $Z(v)$ be the set of nodes that occupied the cache during the interval $[t - c\log N, t]$. For any $\delta > 0$ and sufficiently large constant c, w.h.p. $|Z(v)|$ is in the range $\frac{2Dc}{(C-D)K}\log N(1 \pm \delta)$*

Proof: As in the proof of Lemma 4.3.1, the expected number of connections to a given cache node in an interval $[t - c\log N, t]$ is $\frac{2Dc\log N}{K}$. Applying the Chernoff bound we show that w.h.p. the number of connection is in the range $\frac{2Dc}{K}\log N(1 \pm \delta)$. Since a cache node receives $C-D$ connections while in the cache the result follows. □

The following lemma shows with that in most cases the algorithm finds a replacement node for the cache by searching only a few $O(\log N)$ nodes.

Lemma 4.3.3 *Assume that $C > 3D$. At any time $t \geq c\log N$, with probability $1 - O(\frac{\log^2 N}{N})$ the algorithm finds a replacement d-node by examining only $O(\log N)$ nodes.*

Proof: Let $v_1, ..., v_K$ be the K nodes in the cache at time t. With probability

$$Ke^{-\frac{c\log^2 N}{N}} \geq 1 - O(\frac{\log^2 N}{N})$$

no node in $Z(v_i)$, $i = 1, .., K$ leaves the network in the interval $[t - c\log N, t]$.

Suppose that node v leaves the cache at time t, then the protocol tries to replace v by a d-node neighbor of a node in $Z(v)$. As in the proof of Lemma 4.3.1 w.h.p. $Z(v)$ received at least $\frac{2D}{K}c\log N$ connections from new d-nodes in the interval $[t - c\log N, t]$. Among these new d-nodes no more than $|Z(v)|$ nodes enter the cache and became c-nodes during this interval. Using the bound on $|Z(v)|$ from Lemma 4.3.2, w.h.p. there is a d-node attached to a node of $Z(v)$ at time t. □

4.3.3 Connectivity

The proof that at any given time the network is connected w.h.p. is based on two properties of the protocol: (1) Steps 3-4 of the protocol guarantee (deterministically) that at any given time a

node is connected through "preferred connections" to a cache node; (2) The random choices of new connections guarantee that w.h.p. the $O(\log N)$ neighborhoods of any two cache nodes are connected to each other. In Section 4.4 we show that the first component is essential for connectivity. Without it, there is a constant probability that the graph has a number of small disconnected components.

Lemma 4.3.4 *At all times, each node in the network is connected to some cache node directly or through a path in the network.*

Proof: It suffices to prove the claim for c-nodes since a d-node is always connected to some c-node. A c-node v is either in the cache, or it is connected through its preferred connection to a node that was in the cache after v left the cache. By induction, the path of preferred connections must lead to a node that is currently in the cache. □

Lemma 4.3.5 *Consider two cache nodes v and u at time $t \geq c \log N$, for some fixed constant $c > 0$. With probability $1 - O(\frac{\log^2 N}{N})$ there is a path in the network at time t connecting v and u.*

Proof: Let $Z(v)$ be the set of nodes that occupied the cache during the interval $[t - c \log N, t]$. By Lemma 4.3.2, w.h.p. $|Z(v)| = d \log N$, for some constant d.

The probability that no node in $Z(v)$ leaves the network during the interval $[t - c \log N, t]$ is

$$e^{-\frac{cd \log^2 N}{N}} \geq 1 - O(\frac{\log^2 N}{N}).$$

Note that if no node in $Z(v)$ leaves the network during this interval then all nodes in $Z(v)$ are connected to v by their chain of preferred connections.

The probability that no new node that arrives during the interval $[t - c \log N, t]$ connects to both $c(v)$ and $c(u)$ is bounded by $(1 - D^2/K^2)^{c \log N} = O(1/N^{c'})$. □

Since there are $K = O(1)$ cache locations we have the following theorem.

Theorem 4.3.3 *There is a constant c such that at any given time $t > c \log N$,*

$$Pr(G_t \text{ is connected}) \geq 1 - O(\frac{\log^2 N}{N}).$$

The above theorem does not depend on the state of the network at time $t - c \log N$. It therefore shows that the network rapidly recovers from fragmentation.

Corollary 4.3.1 *There is a constant c such that if the network is disconnected at time t,*

$$Pr(G_{t+c \log N} \text{ is connected}) \geq 1 - O(\frac{\log^2 N}{N}).$$

Theorem 4.3.4 *At any given time t such that $t/N \to \infty$, if the graph is not connected then it has a connected component of size $N(1 - o(1))$.*

Proof: By Lemma 3.4 all nodes in the network are connected to some cache node. The $O(\frac{\log^2 N}{N})$ failure probability in Theorem 4.3.3 is the probability that some cache node is left with fewer than $d \log N$ nodes connected to it. Excluding such cache nodes all other cache nodes are connected to each either with probability $1 - K^2(1 - D^2/K^2)^{c \log N} = 1 - 1/N^c$, for some $c > 0$. □

4.3.4 Diameter

We state our main theorem which gives a bound on the diameter of the network. We note that this is the best bound possible (upto constant factors) for a constant degree network.

Theorem 4.3.5 *For any t, such that $t/N \to \infty$, w.h.p. the largest connected component of G_t has diameter $O(\log N)$. In particular, if the network is connected (which has probability $1 - O(\frac{\log^2 N}{N})$) then w.h.p. its diameter is $O(\log N)$.*

Proof: Since a d-node is always connected to a c-node it is sufficient to discuss the distance between c-nodes. Thus, in the following discussion all nodes are c-nodes. For the purpose of the proof we define a constant f, and call a cache node *good* if during its time in cache it receives a set of $r \geq f$ connections such that

- The r connections are "reconnect" connections.
- The r connections are not preferred connections .
- The r connections resulted from r different nodes leaving the network.

We color the edges of the graph using three colors: A, $B1$ and $B2$. All edges are colored A except a random f edges of the the set of r "reconnect" edges that satisfied the three requirements of a good node. A random half of these are colored $B1$, the rest are colored $B2$.

Since the proof of Theorem 4.3.3 use only preferred connection edges, and edges of new d-nodes, it is easy to verify that at any time t, the network is connected with probability $1 - O(\frac{\log^2 N}{N})$ using only A edges, and that if the network is not connected then w.h.p. the A edges define a connected component of size $N(1 - o(1))$.

We rely on the "random" structure of the B edges to reduce the diameter of the network. However, we need to overcome two technical difficulties. First, although the B edges are "random", the occurrences of edges between pairs of nodes are not independent as in the standard $G_{n,p}$ random graph model ([13]). Second, the total number of B edges is relatively small; thus the proof needs to use both the A and the B edges.

Lemma 4.3.6 *Assume that node v enters the cache at time t, where $t/N \to \infty$. Then for a sufficiently large choice of the constant C, the probability that v leaves the cache as a good node is at least $\gamma > 1/2$. Further, the f re-colored edges of a good cache node are distributed almost uniformly at random among the nodes currently in the network. More precisely, the probability that an old node in the network is an endpoint of any of these f connections is $\frac{f}{N} \pm o(\frac{1}{N})$ Furthermore, the probability that a c-node is good is independent of other c-nodes.*

Proof: Consider the interval of time in which v was a cache node.

1. New nodes join the network according to a Poisson process with rate 1. Also the expected number of connections to v from a new node is D/K.

2. Since w.h.p. there are $N(1+o(1))$ nodes in the network at that time, nodes leave the network according to a Poisson process with rate $1-o(1)$ (this can be shown using definition 2.2.3). Also the expected number of connections to v as a result of a old node leaving the network is

$$\sum_{u \in V} \frac{d(u) - cn(u)}{|V|} \frac{D}{d(u)} \frac{1}{K} = \frac{D}{K}(1-o(1)) < 1.$$

where $cn(u)$ is either 1 or 0 depending on whether u is a c-node or not. (If u is a c-node then we don't count the preferred node.)

3. When an old node u leaves the network, the expected number of reconnect edges to v from u is $\frac{d(u)-cn(u)}{N} \frac{D}{d(u)} = \Theta(D/N)$, where $cn(u)$ is defined as above.

From 1 and 2 above, it follows from (near-)symmetry that each connection to v , while it is in the cache, has a constant probability each of being from a new or a old node. Also from 2, we have the expected number of connections to v as a result of one old node leaving the network is ≤ 1; thus each connection has a constant probability of being triggered by a unique node leaving the network. Thus, for a sufficiently large C, the $C-D$ connections to v include, with probability $\gamma > 1/2$, $r \geq f$ re-connect edges from different nodes leaving the network.

Further, from 3 and using the fact that each node leaves the network independently and identically via the same exponential distribution it follows that each node in the network - irrespective of its degree - has an almost equal probability of being connected to v i.e., $\frac{f}{N} \pm o(\frac{1}{N})$. Finally, it is easy to see the independence of the events for different c-nodes, since a cache node stays in the cache till it accepts C connections irrespective of other cache nodes. □

For the proof of the next lemma we need the following definitions. Given a node v in G_t, let $\Gamma_0(v)$ be an arbitrary cluster of $d \log N$ c-nodes, such that $v \in \Gamma_0(v)$, and this cluster has diameter $O(\log N)$ using only A edges. For $i \geq 1$, i odd (resp., even) let $\Gamma_i(v)$ be all the c-nodes in G_t that are connected to $\Gamma_{i-1}(v)$ and are not in $\cup_{j=0}^{i-1} \Gamma_j(v)$ using $B1$ (resp., $B2$) edges.

We first show the following "expansion" lemma which states that each neighborhood of v starting from $\Gamma_1(v)$ is at least twice the size of the previous neighborhood.

Lemma 4.3.7 *If* $|\Gamma_{i-1}(v)| = o(N)$,

$$Pr\{|\Gamma_i(v)| \geq 2|\Gamma_{i-1}(v)|\} \geq 1 - 1/N^5.$$

Proof: Let $W = \Gamma_{i-1}(v)$, $w = |W|$, and let $z \notin W \cup (\cup_{j=0}^{i-1} \Gamma_j(v))$. W.l.o.g. assume that $i-1$ is even. Partition W into W_0, consisting of nodes in W that are older than z, and W_1, consisting of nodes in W that arrived after z. The probability that z is connected to W_0 using $B1$ edges is $\frac{1}{4}\frac{f|W_0|}{N}(1-o(1))$ using lemma 4.3.6. Similarly, each node in W_1 has probability $\frac{f}{4N}(1-o(1))$ of being connected to z by $B1$ edges. Thus, the probability that z is connected to W is at least $\frac{1}{2}\frac{fw}{N}(1-o(1))$.

Let $Y = |\Gamma_i(v)|$ be the number of c-nodes outside W that are connected to W by $B1$ edges. $E[Y] = \frac{f}{4}w(1-o(1))$. Let $w_1, w_2, \ldots$ be an enumeration of the nodes in W, and let $N(w_i)$ be the set of neighbors of w_i outside W using $B1$ edges. Define an exposure martingale $Z_0, Z_1, \ldots$, such

that $Z_0 = E[Y]$, $Z_i = E[Y \mid N(w_1), ..., N(w_i)]$, $Z_w = Y$. Since the degree of all nodes is bounded by C, a node w_i can connect to no more than C nodes outside W. Thus, $|Z_i - Z_{i-1}| < C$.

Using Azuma's inequality [8] it follows that that for sufficiently large constant d,

$$Pr\{|Y - E[Y]| \geq \frac{f}{8}\frac{\sqrt{w}}{C}C\sqrt{w}\} \leq 2e^{-\frac{f^2}{128C^2}w} \leq 1/N^5.$$

□

Now we complete the proof of Theorem 4.3.5. Our goal is to show that w.h.p the distance between any two c-nodes is $O(\log N)$. Consider any two c-nodes v and u. By applying lemma 4.3.7 repeatedly $O(\log N)$ times we have with probability $1 - O(\frac{\log N}{N^5})$, for some $k_v, k_u = O(\log N)$, $|\Gamma_{k_v}(v)| \geq \sqrt{N}\log N$ and $|\Gamma_{k_u}(u)| \geq \sqrt{N}\log N$. The probability that $\Gamma_{k_v}(v)$ and $\Gamma_{k_u}(u)$ are disjoint and not connected by an edge is bounded by $(1 - f/2N)^{N\log^2 N}$, thus with probability $1 - O(\frac{\log N}{N^5})$ an arbitrary pair of nodes u and v are connected by a path of length $O(\log N)$ in G_t. Summing the failure probability over all $\binom{n}{2}$ pairs it follows that w.h.p. any pair of nodes in G_t is connected by a path of length $O(\log N)$. □

4.4 Why Preferred Connections?

In this section we show that the preferred connection component in our protocol is essential: running the protocol without it leads to the formation of many small disconnected components. A similar argument would work for other fully decentralized protocols that maintain a minimum and maximum node degree and treat all edges equally, i.e., do not have preferred connections. Observe that a protocol cannot replace all the lost connections of nodes with degree higher than the minimum degree. Indeed, if all lost connections are replaced and new nodes add new connections, then the total number of connections in the network is monotonically increasing while the number of nodes is stable, thus the network cannot maintain a maximum degree bound.

To analyze our protocol without preferred nodes define a type H subgraph as a complete bipartite network between D d-nodes and D c-nodes, as shown in Figure 4.1.

Lemma 4.4.1 *At any time $t \geq c$, where c is a sufficiently large fixed constant, there is a constant probability (i.e. independent of N) that there exists a subgraph of type H in G_t.*

Proof: A subgraph of type H arises when D incoming d-nodes choose the same set of D nodes in cache. A type H subgraph is present in the network at time t when all the following four events happen:

1. There is a set S of D nodes in the cache each having degree D (i.e., these are the new nodes in the cache and are yet to accept connections) at time $t - D$.

2. There are no deletions in the network during the interval $[t - D, t]$.

3. A set T of D new nodes arrive in the network during the interval $[t - D, t]$.

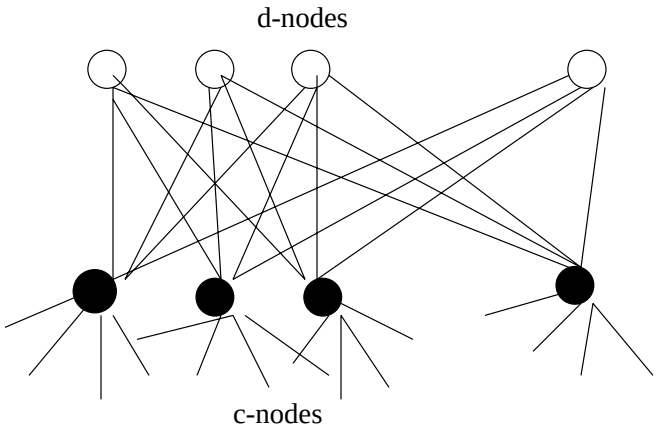

Figure 4.1: Subgraph H used in proof of lemma 4.4.2. Note that $D = 4$ in this example. All the four d-nodes are connected to the same set of four c-nodes (shown in black).

4. All the incoming nodes of set T choose to connect to the D cache nodes in set S.

Since each of the above events can happen with constant probability, the lemma follows. □

Lemma 4.4.2 *Consider the network G_t, for $t > N$. There is a constant probability that there exists a small (i.e., constant size) isolated component.*

Proof: By Lemma 4.4.1 with constant probability there is a subgraph (call it F) of type H in the network at time $t-N$. We calculate the probability that the above subgraph F becomes an isolated component in G_t. This will happen if all $2D$ nodes in F survive till t and all the neighbors of the nodes in F (at most $C(C-D)$ of them connected to the D c-nodes) leave the network and there are *no* re-connections. The probability that the $2D$ subgraph nodes survived the interval $[t-N, t]$ is e^{-2D}. The probability that all neighbors of the subgraph leave the network with no new connections is at least $(1-e)^{-C(C-D)}(1-\frac{D}{D+1})^{C(C-D)}$. Thus, the probability that F becomes isolated is at least

$$e^{-2D}(1-e)^{-C(C-D)}(1-\frac{D}{D+1})^{C(C-D)} = \Theta(1)$$

□

Theorem 4.4.1 *The expected number of small isolated components in the network at any time $t > N$ is $\Omega(N)$, when there are no preferred connections.*

Proof: Let S be the set of nodes which arrived during the interval $[t-N, t-\frac{N}{2}]$. Let $v \in S$ be a node which arrived at at t'. From the proof of Lemma 4.4.2 it is easy to show that v has a constant probability of belonging to a subgraph of type H at t'. Also, by the same lemma, H has a constant probability of being isolated at t. Let the indicator variable X_v, $v \in S$ denote the probability that v belongs to a isolated subgraph at time t. Then, $E[\sum_{v \in S} X_v] \geq \Omega(N)$, by linearity of expectation. Since the isolated subgraph is of constant size, the theorem follows. □

4.5 Related Work

We briefly discuss related work in P2P systems most relevant to our work. Two important systems proposed recently are Chord [74] and CAN [66]. These are content-addressable protocols i.e., they solve the problem of efficiently locating a node storing a given data item. There are two components for the above protocols: the first specifies how and where a particular data item should be stored in the network, and the second specifies a routing protocol to retrieve a given data item efficiently.

The focus of our work is building P2P networks with good topological properties and not the problem of searching or routing – which is an orthogonal issue for us; for example a Gnutella-like [39] or a Freenet-like [22] search/routing mechanism can be easily incorporated in our protocol. Thus, although we cannot directly compare our protocol with content-addressable networks such as Chord or CAN, we can compare them with respect to their topological properties and guarantees. CAN uses a d-dimensional Cartesian coordinate space (for some fixed d) to implement a distributed hash table that maps keys onto values. Chord on the other hand, uses a scheme called *consistent hashing* to map keys to nodes. Although the degree (the number of entries in the routing table of a node) of CAN is a fixed constant d (the number of entries in its routing table), the diameter (the maximum distance between any two nodes in the virtual network) can be as large as $O(dn^{1/d})$. In the case of Chord, the diameter is $O(\log N)$ while the degree of every node is $O(\log N)$. (If $d = \log N$), CAN matches the bounds of Chord.) This is in comparison to the constant degree and logarithmic diameter of our protocol. However, the most important contrast is that their protocols provide no provable guarantees in a realistic dynamic setting, unlike ours. Chord gives guarantees only under a simplistic assumption that every node can fail (or drop out) with probability $1/2$.

Another interesting P2P system is the dynamically fault-tolerant network of [70]. This is again a content-addressable network based on a butterfly topology. The diameter of the network is $O(\log N)$ and the degree is $O(\log^3 N)$. Peer insertion takes $O(\log N)$ time. The system is robust to fault tolerance in the sense that at any time, an arbitrarily large fraction of the peers can reach an arbitrarily large fraction of the data items. They show the above property under a somewhat artificial assumption that in any time interval during which an adversary deletes some number of peers, some larger number of peers join the network. Also they assume that each of the new peers joining the network knows one *random* peer currently in the network. To compare with our work, we show that our protocol is naturally fault-tolerant (in the sense it recovers fairly rapidly from fragmentation and high diameter with high probability) under a natural dynamic model where each node operates with no global knowledge.

4.6 Discussion and further work

We give a simple distributed local P2P protocol to construct networks with good topological properties - namely constant degree, connectivity and low-diameter. We analyze our protocol under a realistic dynamic setting and prove rigorously that it results in the above properties with large

probability. We also proved that our protocol is naturally robust to failures is that it has nice self-correcting properties such as rapid recovery from network fragmentation. We now discuss possible extensions and future work.

It is important to point out our protocol is concerned with building a good *virtual* network topology which may not match the underlying Internet topology (of course, this may not be a big issue for enterprise P2P). In fact, evidence [67] suggests that these two topologies do not match well. It will be of practical interest [67] to construct topologies that respects the underlying physical topology (e.g., locality) - this an area for further research.

In our protocol we implicitly assume that all nodes have equal capabilities (i.e., storage and number of connections supported) and all links have equal bandwidth. In enterprises with homogeneous systems this is closer to reality, however this is not the case in the Internet. It will be nice to extend our protocol to incorporate heterogeneous nodes and links.

Chapter 5

Online Computation

In this chapter, we study performance of online algorithms under a very general stochastic input model. This focuses on an important aspect of online computation that is not addressed by the standard competitive analysis. Namely, identifying request sequences for which non-trivial online algorithms are useful versus request sequences for which all algorithms perform equally bad.

Our approach in this work is based on the relation between entropy, compression and gambling, extensively studied in information theory. It has been shown that in some settings entropy can either fully or at least partially characterize the expected outcome of an iterative gambling game. Viewing online problem with stochastic input as an iterative gambling game, our goal is to study the extent to which the entropy of the input characterizes the expected performance of online algorithms for problems that arise in computer applications. We study bounds based on *entropy* for three important online problems – list accessing, prefetching and caching. We show that entropy is a good performance characterizer for prefetching, but not so good characterizer for online caching. Our work raises several open questions in using entropy as a predictor in online computation.

5.1 Introduction

Advanced system and architecture design allows dynamic allocations of resources to online tasks such as prefetching and caching. To fully utilize this feature the system needs an efficient mechanism for estimating the expected gain from using these resources. Prefetching, for example, is an expensive operation since it "burns instruction bandwidth"[42]. However, successful prefetching can significantly speedup computation. Thus, one needs to compare the gain from prefetching on a given data stream to the cost in instruction bandwidth. The tradeoff between resource allocation and gain is even more transparent in the case of malleable cache [55, 75, 25]. In this architecture the cache can be dynamically partitioned between different data streams. A data stream that can make better use of a larger cache is assigned more space, while a stream with very little structure or repeats is allocated a smaller cache space. Again, efficient utilization of this technology requires a mechanism for predicting caching gain for a given data stream.

Online algorithms have been studied in the theory community mainly in the context of *competitive analysis* (see [16] for a comprehensive survey). Competitive analysis compares the performance of different algorithms, but it gives no information about the actual gain from using them. In particular, even the best algorithm under the competitive analysis measure might fail on almost all requests of some sequence. Thus, an entirely new approach is needed in order to quantify the amount of resources the system should allocate to a given online process. In this work we explore the relation between the entropy of the stream of requests and the gain expected from online algorithm performing on this request sequence. Entropy measures the randomness or uncertainty of a random process. We expect online algorithms to perform well on highly predictive request sequences, generated by a source with low entropy, and to perform poorly on sequences with little pattern, generated by a high entropy source. Our work is motivated by the extensive work in information theory relating data compression, entropy and gambling. It has been shown that for some special cases of gambling games the entropy of the stochastic process fully characterizes the maximum expected profit for any strategy for that game (see section 5.1.1). Our goal is to explore similar relations between entropy and online problems in computer applications. We discuss here three online problems: list accessing, prefetching and caching. We show that in the case of prefetching entropy gives a good characterization of the best online performance that is possible, while in the case of caching entropy does not fully characterize the best online performance.

5.1.1 Related Work

The three online problems considered here were extensively studied in the competitive analysis model. It has been shown in [72] that the competitive ratio [1] of the *move to front (MTF) algorithm* for the list accessing problem is two [72]. In the case where the input sequence is drawn from a discrete memoryless source the MTF algorithm has been compared to the performance of a static offline algorithm SOPT that initially arranges the list in decreasing order of request probabilities and never reorders them thereafter. It was shown in [36] that $\mathrm{MTF}(D) \leq \frac{\pi}{2}\mathrm{SOPT}(D)$, where D is the distribution of the source. Albers et.al [5] analyze the performance of the TIMESTAMP algorithm on a discrete memoryless source with distribution D and proved that for any distribution D, $\mathrm{TIMESTAMP}(D) \leq 1.34 \times \mathrm{SOPT}(D)$ and with high probability, $\mathrm{TIMESTAMP}(D) \leq 1.5 \times \mathrm{OPT}(D)$. The actual work done by the MTF algorithm was studied when the request sequence is generated by a discrete memoryless source with probability distribution D[5, 12, 36].

For online caching (or demand paging) the well known LRU (Least Recently Used) has a competitive ratio of k [72], where k is the cache size, while the randomized MARKER algorithm is $2\log k$ competitive [29]. Franaszek and Wagner [33] studied a model in which every request is drawn from a discrete memoryless source. Karlin et.al [44] study Markov paging where the sequence of page requests is generated by a Markov chain. Their main result is an efficient algorithm which for any Markov chain will achieve a fault-rate at most a constant times optimal.

[1] An online algorithm ALG has a competitive ratio of c if there is a constant α such that for all finite input sequences I, $\mathrm{ALG}(I) \leq c \times \mathrm{OPT}(I) + \alpha$, where OPT is the optimal offline algorithm.

For the problem of prefetching, competitive analysis is meaningless as the optimal offline algorithm will always prefetch the correct item and hence incurs no cost. Vitter and Krishnan [78] consider a model where the sequence of page requests is assumed to be generated by a Markov source (defined in section 5.5), a model which is closest in spirit to (but less general than) our model of a stationary ergodic process. They show that the fault rate of a Ziv-Lempel [81] based prefetching algorithm approaches the fault rate of the best prefetcher (which has full knowledge of the Markov source) for the given Markov source as the page request sequence length $n \to \infty$.

Kelly [47] was the first to study the relation between data compression, entropy and gambling, showing that the outcome of a horse race gambling with fair odds is fully characterized by the entropy of the stochastic process. It was shown [47, 6] that the growth rate of investment in the horse race is equal to $\log m - H$, where m is the number of horses and H is the entropy of the source. Similar results have been shown for portfolio selection strategies in equity market investments [7, 23]. Our results on list accessing are based on the work of Bentley et.al [12] who showed that any list update algorithm can be used to develop a data compression scheme. They also showed that for a discrete memoryless source the expected number of bits needed to encode an alphabet using MTF is linear in the entropy of the source. Similar results have been shown by Albers et.al [5] for the TIMESTAMP algorithm. Our results on prefetching are motivated by the work of Feder and Merhav [30] relating entropy of a discrete random variable to the minimal attainable probability of error in guessing its value. In the context of prefetching their results can be viewed as giving a tight bound on the fault rate when the size of cache is 1. A tight lower bound on this error probability is given by Fano's inequality [23, theorem 2.11.1]. Their main result is a tight upper bound for the fault rate when $k = 1$. Feder and Merhav also showed that the same lower and upper bounds (for $k = 1$) hold for a stationary ergodic source. However, their upper bound does not seem to generalize to higher values of k. Note that there is more work in the information theory on predicting binary sequences (corresponding to prefetching in a universe of two pages with cache of size 1) [31], however these results cannot be generalized to our prefetching scenario. Our approach to deriving the upper bound on the fault rate for an arbitrary ergodic source and arbitrary cache size k is different and is based on the well-known Liv Zempel universal algorithm for data compression [81]. Our proof uses Rissanen's interpretation of the Liv Zempel Algorithm [68]. See Algoet [6] for further results on universal schemes for prediction, gambling and portfolio selection.

5.1.2 New Results

We focus on three online problems: list accessing, prefetching, and caching. Our goal is to study the relation between the entropy of the sequence of requests and the best performance of an online algorithm for these problems.

We assume that the sequence of requests is generated by a *discrete stationary ergodic process* [34, definition 3.5.13] which is the most general stochastic source considered in information theory.

For the list accessing problem we show that any deterministic online algorithm requires an average work of $\Omega(2^H)$ steps per item access, where H is the entropy of the input source.

For the prefetching problem we give an upper and lower bound showing that the average number of faults of the best algorithm is linear in H, the entropy of the input source. Our lower bound on the fault rate can be seen as a generalization of Fano's inequality for $k > 1$. Our upper bound generalizes the well known upper bound of $\frac{1}{2}H$ on the minimal error probability for guessing the value of a discrete random variable (i.e. $k = 1$) shown by completely different techniques [34, pages 520-521], [30, 41].

In contrast, we consider the caching problem for two stationary ergodic sources with equal entropy. We show that the best caching fault rate for the two sources fall in two disjoint intervals as a function of H, the entropy of the source. Thus, in the case of caching, entropy alone is not a sufficient predictor for online performance.

5.2 A General Stochastic Model

We model a request sequence for an online process as a *stochastic process*, denoted by $\{X_i\}$. The range of $\{X_i\}$, for all i is the finite alphabet set $\mathcal{H}$. Let $l = |\mathcal{H}|$. In this work, we restrict ourselves to stochastic sources which are both *stationary* and *ergodic*. This is a very general model and includes powerful models such as memoryless and (stationary) Markov sources [34, 78, 24]. This type of source also has many nice properties which make them amenable to analysis.

We would like to characterize the performance of the best online algorithm based on the *entropy of the request sequence*, which is the *entropy rate* of the stochastic process (or the source) generating the sequence.

A stochastic process is stationary if its probabilistic description is time invariant. Stationarity property guarantees the existence of a well-defined entropy rate, denoted by H (or H_n for a sequence of length n). Ergodicity is a technical condition, which implies that a source cannot be separated into persistent modes of behavior. Formally, it means that the law of large numbers holds for such a source. See chapter 2 for formal definitions of the above terms and other basic concepts from information theory. Throughout this chapter, we use the notation lg to denote logarithm to the base 2.

5.3 List Accessing

We start with a simple example relating the cost of online list accessing to the entropy of the request sequence. As in Borodin & El-Yaniv [16] we consider the *static list accessing model* in which a fixed set of l items, is stored in linked list. The algorithm has to access sequentially a sequence of n requests for items in the list. The access cost $a(X_i)$ of an item is the number of links traversed by the algorithm to locate the item, starting at the head of the list. Before each access operation the algorithm can rearrange the order of items in the list by means of transposing an element with an adjacent one. The cost is 1 for a single exchange. Let $c(X_i)$ be the total cost associated with servicing element X_i. $c(X_i)$ includes both the access cost $a(X_i)$ and any transposition cost incurred

before servicing X_i.

Following Bentley et al. [12] we explore the relation between list accessing and data compression by using the linked list as a data structure of a data compression algorithm. Assume that a sender and a receiver start with the same linked list, and use the same rules for rearranging the list throughout the execution. Instead of sending item X, the sender needs only to send the distance i of X from the head of the linked list, i.e. the work involved in retrieving item X. We encode the integer distance by using a variable length prefix code. The lower bound depends on the particular encoding used for the distance. Consider an encoding scheme that encodes an integer i using $f(i)$ bits. To get a lower bound on the work done, we need f to be a concave nondecreasing function (when defined on the non-negative real).

Theorem 5.3.1 *Let $\bar{c}$ be the average cost of accessing an item by any deterministic algorithm $\mathcal{A}$ on a stationary ergodic sequence of requests $< X >= X_1, X_2, \ldots, X_n$. Then $\bar{c} \geq f^{-1}(H)$, where H is the entropy of the sequence, and f is a concave nondecreasing invertible function such that there is an encoding scheme for the integers that encodes integer i with up to $f(i)$ bits.*

Proof: $\bar{c} \geq \frac{1}{n}\sum_{i=1}^{n} c(X_i)$, and $c(X_i) \geq a(X_i)$, where $a(X_i)$ is the distance of X_i from the head of the linked list at time i, which is the value sent by the sender at time i. If the sender encodes $a(X_i)$ using $f(a(X_i))$ bits, then by variable-length source coding theorem [34, theorem 3.5.2] and by equations 2.16 to 2.19,

$$\frac{1}{n}\sum_{i=1}^{n} f(a(X_i)) \geq H_n \geq H \tag{5.1}$$

Since f is concave, by Jensen's inequality and using 5.1,

$$f(\bar{c}) \geq f(\frac{1}{n}\sum_{i=1}^{n} a(X_i)) \geq \frac{1}{n}\sum_{i=1}^{n} f(a(X_i)) \geq H$$

Hence, $\bar{c} \geq f^{-1}(H)$. □

We can now get concrete lower bounds by plugging in appropriate coding functions. A simple prefix coding scheme encodes an integer i using $1 + 2\lfloor \lg i \rfloor$ bits [2] [27]. The encoding of i consists of $\lfloor \lg i \rfloor$ 0's followed by the binary representation of i which takes $1 + \lfloor \lg i \rfloor$ bits, the first of which is a 1. This encoding function gives the following corollary to theorem 5.3.1.

Corollary 5.3.1 *Any deterministic online algorithm for list accessing has to incur an average cost of $2^{(H-1)/2}$ per item, where H is the entropy rate of the sequence.*

We get a better lower bound by replacing the $\lfloor \lg i \rfloor$ 0's followed by a 1 in the above scheme by $\lg\lfloor 1 + \lg l \rfloor$ bits giving an encoding for i with $\lfloor \lg i \rfloor + \lg\lfloor 1 + \lg l \rfloor$ bits. Using this scheme we prove:

[2] This function can be made invertible in the obvious way to obtain the lower bound specified in corollary 5.3.1. Similar comment holds for the encoding function used to derive corollary 5.3.2.

Corollary 5.3.2 *The average cost of accessing an item for a deterministic online algorithm is at least* $\frac{2^H}{\lceil \lg l+1 \rceil}$, *where* l *is the size of the alphabet.*

We note that theorem 5.3.1 actually applies for any list accessing algorithm even if it is a randomized algorithm. That is, for a randomized list accessing algorithm the expected cost of accessing an item is lower bounded as specified by theorem 5.3.1. This follows from Yao's Minimax Principle [56]. Thus the lower bounds derived in the corollaries hold for randomized algorithms also.

We conclude our discussion on list accessing by showing a lower bound on the performance of the well-studied static offline algorithm (SOPT) on a discrete memoryless source with probability distribution $D = \{p_1, \ldots, p_l\}$ where p_i is the probability of accessing item x_i. SOPT initially arranges the list in decreasing order of request probabilities and never reorders them thereafter. Although SOPT is not an online algorithm, because of its simplicity has been used as a benchmark for comparison to other (online) algorithms [16]. The expected access cost per item is given by $\mathrm{SOPT}(D) = \sum_{i=1}^{l} ip_i$. The following theorem gives a lower bound on the expected cost of accessing an item by SOPT based on the entropy of the input distribution D.

Theorem 5.3.2 *The expected cost of accessing an item by SOPT on a discrete memoryless source with distribution* D *is at least* 2^{H-1} *where* H *is the entropy of* D.

Proof: Suppose let w (an integer for now) be the average cost of accessing an item by SOPT. Note that $w \leq (l+1)/2$. Then the probability distribution which maximizes the entropy is given by $(\underbrace{\frac{1}{2w-1}, \ldots, \frac{1}{2w-1}}_{2w-1 \ terms}, \underbrace{0, \ldots, 0}_{(l-(2w-1)) \ terms})$ and the corresponding entropy is given by $\lg(2w-1)$. This can be shown by a straightforward verification of the Kuhn-Tucker conditions for optimality of the following non-linear program:

$$\text{Maximize} \quad -\sum_{i=1}^{l} p_i \lg p_i$$

subject to the following constraints:

$$\sum_{i=1}^{l} ip_i = w$$
$$\sum_{i=1}^{l} p_i = 1$$
$$p_i \geq 0,\ i = 1, \ldots, l$$
$$p_i - p_{i+1} \geq 0,\ i = 1, \ldots, l-1$$

Thus, if H is the entropy of any distribution D then, $H \leq \lg(\lceil 2w-1 \rceil)$, which yields the result. □

5.4 Prefetching

As in [78] we consider the following formalization of the prefetching problem: we have a collection $\mathcal{H}$ of pages in memory and a cache of size k, and typically $k \ll |\mathcal{H}|$. The system can prefetch k items to the cache prior to each page request. The fault rate is the average number of steps in which the requested item was not in the cache.

Let $l = |\mathcal{H}|$. Given a request sequence $< X >= X_1, X_2, \ldots X_n$, we are interested in the *minimal expected page fault rate* of a request sequence i.e., the minimum long term frequency of page faults that is possible for the sequence. We show the existence of this quantity when the request sequence is generated by a stationary ergodic process.

5.4.1 Lower Bound

We first prove the lower bound for a discrete memoryless source, generalizing the result in Feder and Merhav [30].

We observe that the optimal prefetching strategy in a discrete memoryless source is obvious (a consequence of Bayes' decision rule, for example see [41]):

Lemma 5.4.1 *Let $p(.)$ be a probability distribution on $\mathcal{H}$. Suppose each page in the sequence is drawn i.i.d with probability distribution $p(.)$. Then the minimal expected page fault rate can be obtained by picking the pages (in the cache) with the top k probabilities. Hence the minimal expected fault rate is given by $1 - \sum_{x \in T(p(.))} p(x)$, where $T(p(.))$ is the set of pages with the top k probabilities in the distribution $p(.)$.*

Our goal is to relate the fault rate of the above strategy to the entropy of the source. Consider a discrete random variable X, and let $p(i) = Pr\{X = i\}$ for $i \in \mathcal{H}$. Assume without loss of generality that $p(1) \geq p(2) \geq \cdots \geq p(l)$. Let $P = [p(1), \ldots, p(l)]$ be the probability vector and let $P_\pi = \{P \mid p(i) \geq 0, \ \forall i, \ \sum_{i=1}^{l} p(i) = 1 \text{ and } \sum_{i=1}^{k} p(i) = 1 - \pi\}$ Let $H(P)$ (or $H(X)$) be the entropy of the random variable having the distribution given by P. Given the minimal expected fault rate $\pi(X)$ (or π for simplicity) we would like to find a upper bound on the entropy as $H(X) \leq \max_{P \in P_\pi} H(P)$.

Lemma 5.4.2 *Let the minimal expected page fault rate be π. Then the maximum entropy $H(P_{max}(\pi))$ is given by $(1-\pi)\lg(\frac{k}{1-\pi}) + \pi \lg(\frac{l-k}{\pi})$.*

Proof: Given the minimal expected page fault rate π, the maximum entropy distribution $P_{max}(\pi)$ is given by $(\underbrace{\frac{1-\pi}{k}, \ldots, \frac{1-\pi}{k}}_{k \ terms}, \underbrace{\frac{\pi}{l-k}, \ldots, \frac{\pi}{l-k}}_{(l-k) \ terms})$
assuming $\pi \leq 1 - k/l$ (which is always true). This distribution maximizes the entropy because of the following argument. Let $p(x)$ be any probability distribution on $\mathcal{H}$. Then the relative entropy (or Kullback Leibler distance) between $p(x)$ and $P_{max}(\pi)$ is given by [23, definition 2.26]

$$\sum_{x \in \mathcal{H}} p(x) \lg(p(x)/P_{max}(\pi)) =$$
$$-H(X) + \sum_{x \in \mathcal{H}} p(x) \lg 1/P_{max}(\pi))$$

Since the relative entropy is always positive[23, Theorem 2.6.3] we have

$$H(X) \leq \lg(\frac{k}{1-\pi}) \sum_{x=1}^{k} p(x) + \lg(\frac{l-k}{\pi}) \sum_{x=k+1}^{l} p(x) = (1-\pi)\lg(\frac{k}{1-\pi}) + \pi \lg(\frac{l-k}{\pi})$$

□

Corollary 5.4.1 $\pi \geq \frac{H-1-\lg k}{\lg(\frac{l}{k}-1)}$

Proof: From lemma 5.4.2,

$$H \leq -(1-\pi)\lg(1-\pi) - \pi \lg \pi + (1-\pi)\lg k + \pi \lg(l-k) = h(\pi) + (1-\pi)\lg k + \pi \lg(l-k)$$

where, $h(\pi) = -\pi \lg \pi - (1-\pi)\lg(1-\pi)$ is the binary entropy function which takes values between 0 and 1. Hence, $H \leq 1 + \lg(k^{1-\pi}(l-k)^{\pi})$ which gives the result. □

We now show that the same lower bound holds for any stationary ergodic process generalizing the argument of [30, Theorem 1]. First we need to define the following. Let (X,Y) be a pair of discrete random variables (each with range $\mathcal{H}$) with joint distribution $p(x,y)$. For the following let $T(.)$ be defined as in 5.4.1. Then by lemma 5.4.1 the minimal expected fault rate that can be obtained (using a cache of size k) given that a page y of Y was observed is

$$\pi(X|Y) = \sum_{y}[1 - \sum_{x \in T(p(.|y))} p(x|y)]p(y) = \sum_{y} \pi(X|Y=y)p(y) \tag{5.2}$$

Let $\{X_i\}_{i=1}^{\infty}$ be a stationary ergodic process. Similar to (see equation 2.16) the entropy of a stationary process we define the fault rate of a stationary ergodic sequence as

$$\Pi(\mathcal{H}) = \lim_{n->\infty} \pi(X_n|X_{n-1}, \ldots, X_1) \tag{5.3}$$

To show that the above limit exists, we need the following lemma which shows that conditioning cannot increase expected minimal fault rate.

Lemma 5.4.3 *Let (X,Y) be a pair of discrete random variables as defined above. Then,* $\Pi(X|Y) \leq \Pi(X)$.

Proof:

$$\Pi(X) = 1 - \sum_{x \in T(p(.))} p(x) \tag{5.4}$$

$$\Pi(X|Y) = \sum_{y}(1 - \sum_{x \in T(p(.|y))} p(x|y))p(y) \tag{5.5}$$

where $p(.|y)$ is the conditional probability distribution of X given y. Hence,

$$\begin{aligned}
&\Pi(X) - \Pi(X|Y) = \\
&\sum_y \sum_{x \in T(p(.|y))} p(x|y))p(y) - \sum_{x \in T(p(.))} p(x) \\
&= \sum_y \sum_{x \in T(p(.|y))} p(x,y) - \sum_{x \in T(p(.))} p(x) \\
&\geq \sum_{x \in T(p(.))} \sum_y p(x,y) - \sum_{x \in T(p(.))} p(x) = 0
\end{aligned}$$

□

Lemma 5.4.4 *The limit defined in 5.3 exists for a discrete stationary ergodic process.*

Proof:

$$\begin{aligned}
\Pi(X_{n+1}|X_n, \ldots, X_1) &\leq \Pi(X_{n+1}|X_n, \ldots, X_2) \\
&= \Pi(X_n|X_{n-1}, \ldots, X_1) \qquad (5.6)
\end{aligned}$$

where the inequality follows from the fact that conditioning cannot increase the minimal expected fault rate and the equality follows from the stationarity of the process. Since $\Pi(X_n|X_{n-1}, \ldots, X_1)$ is a non-increasing sequence of non-negative numbers, it has a limit. □

An immediate corollary of the following lemma (in conjunction with equations 2.16 and 5.3) is that the same lower bound as in corollary 5.4.1 holds for stationary ergodic processes too.

Lemma 5.4.5 $\pi(X|Y) \geq \frac{H(X|Y) - 1 - \lg k}{\lg(\frac{l}{k} - 1)}$

Proof: $H(X|Y = y)$ and $\pi(X|Y = y)$ are the entropy and the minimal expected fault rate of a discrete random variable that takes values in $\mathcal{H}$. Thus the lower bound of corollary 5.4.1 holds for every y, i.e.,

$$\begin{aligned}
\pi(X|Y = y) \geq & \quad \frac{H(X|Y=y) - 1 - \lg k}{\lg(\frac{l}{k} - 1)} \\
\Pi(X|Y) = & \quad \textstyle\sum_y \pi(X|Y = y)p(y) \geq \\
& \textstyle\sum_y (\frac{H(X|Y=y) - 1 - \lg k}{\lg(\frac{l}{k} - 1)})p(y) \\
& = \frac{H(X|Y) - 1 - \lg k}{\lg(\frac{l}{k} - 1)}
\end{aligned}$$

□

Thus we can state the following theorem where we have used $\pi(H, k)$ to emphasize the dependence of π on H and k.

Theorem 5.4.1 *The minimal expected page fault rate $\pi(H, k)$ on a request sequence generated by a stationary ergodic process with entropy H is lower bounded by $L(H, k) = \frac{H - 1 - \lg k}{\lg(\frac{l}{k} - 1)}$.*

5.4.2 Upper bound

Our upper bound will use Rissanen's universal data compression system [68] which is a variant of the Ziv-Lempel's universal compression algorithm [81].

The Ziv-Lempel algorithm parses individual sequences $< X^n >= X_1, X_2, \ldots, X_n$ into phrases. Each phrase starts with a comma, and consists of a maximal length sequence that has occurred as an earlier phrase, followed by the next symbol. We denote by v_n the number of complete phrases when parsing the finite sequence $< X^n >$. For example, the binary string $< X^n >= 0101000100$ with length $n = 10$ is parsed as $, 0, 1, 01, 00, 010, 0$ and contains $v_n = 5$ complete phrases and an incomplete phrase at the end. The Ziv-Lempel parsing is obtained by maintaining a dynamically growing tree data structure. Initially this tree consists of a single node, the root. Edges of the tree are labeled with symbols of the alphabet $\mathcal{H}$. Processing of a new phrase starts at the root and proceeds down the tree through edges that match the symbols of the input sequence. When the process reaches a leaf it adds a new branch labeled with the next symbol of the input sequence, which is the last symbol of this phrase. Let T_n denote the tree after processing n symbols of the input.

Rissanen [68] has studied a variant of this algorithm which generates a tree $\tilde{T}_n$. The nodes of T_n are the internal nodes of $\tilde{T}_n$. An internal node of $\tilde{T}_n$ has all its $l = |\mathcal{H}|$ possible descendents. Thus, nodes in $\tilde{T}_n$ are either leaves or have l descendents. Thus, a processing of a phrase in $\tilde{T}_n$ ends when the process reaches a leaf. The leaf is then converted to an internal node, and its l descendents are added to the tree. Note that Rissanen's variant generates exactly the same phrases as the Liv-Zempel parsing. Let v_n be the number of phrases in the parsing of the input string. It is easy to verify that T_n contains $v_n + 1$ nodes, while $\tilde{T}_n$ contains $1 + l(v_n + 1)$ nodes, namely $v_n + 1$ interior nodes and $1 + (l-1)(v_n + 1)$ leaves. The advantage of Rissanen's version is that all leaves in the the tree $\tilde{T}_n$ have equal probability of being reached while searching for a new phrase [68, 6].

Consider the following prefetching algorithm using $\tilde{T}_n$: Assume that at step n the algorithm is at node z of the tree $\tilde{T}_n$. If z is a leaf we prefetch k symbols randomly and go to the root (after making the leaf an interior node and adding l children). If z is an interior node then we prefetch the k items that correspond to the k subtrees, rooted at z, with the maximum number of leaves. When the $(n+1)$th request is revealed the process proceeds through the corresponding branch.

To analyze the above prefetching algorithm we need the following basic results proven by Ziv and Lempel [54, 81].

Theorem 5.4.2 *[54] The number of phrases v_n in a distinct parsing of a sequence (from an alphabet of size l) $X_1, X_2, \ldots, X_n$ satisfies*

$$v_n \leq \frac{n \lg l}{(1-\epsilon_n) \lg n} \quad where \quad lim_{n\to\infty} \epsilon_n = 0$$

Theorem 5.4.3 *[81] Let $\{X_n\}$ be a stationary ergodic process with entropy rate $H(\mathcal{H})$ and let v_n be the number of phrases in a distinct parsing of a sample of length n from this process. Then*

$$limsup_{n\to\infty} \frac{v_n \lg v_n}{n} \leq H(\mathcal{H})$$

We first show a simple upper bound which shows that π is bounded above by a linear function of H.

Theorem 5.4.4 *The minimal expected fault rate $\pi(H,k)$ of the prefetching algorithm on a request sequence generated by a stationary ergodic process with entropy H is upper bounded by $U(H,k) = \frac{(l-k)H}{l\lg(k+1)}$.*

Proof: We assume that $l \geq k+1$, otherwise the fault rate is 0. Since we prefetch the k items corresponding to the k largest subtrees, whenever we incur a fault the symbol corresponds to a branch with at most $1/(k+1)$ leaves of the current subtree. Since the total number of leaves in the completed tree is at most $v_n(l-1)+1$ the number of faults incurred while traversing from the root to a leaf is at most $\lg_{k+1}(v_n(l-1)+1)$. Since all leaves have equal probability, the probability of a fault at a given branch is at most $1-k/l$. Thus, the expected number of faults while processing a phrase is at most $(\frac{l-k}{l})\lg_{k+1}(v_n(l-1)+1)$, and the expected number of faults incurred while processing a sequence of length n is at most

$$\begin{aligned} & \frac{v_n}{n}(\frac{l-k}{l})\lg_{k+1}(v_n(l-1)+1) \\ \leq & \frac{l-k}{l\lg(k+1)}\frac{v_n}{n}(\lg(v_n+1)+\lg l) \\ \leq & \frac{(l-k)H}{l\lg(k+1)} \text{ as } n\to\infty \end{aligned}$$

using theorems 5.4.2 and 5.4.3. □

The above bound shows that the fault rate is bounded above by a linear function of H, although it is weak when $H \geq \lg k$. We can get tighter upper bounds on the fault rate by a more careful analysis. For simplicity we first show the bound for a discrete memoryless source and then we can show that the same upper bound holds for any stationary ergodic source using arguments similar to lemma 5.4.5.

Theorem 5.4.5 *Consider a discrete memoryless source of entropy H and minimal expected fault rate π. Then we have the following bound: $\pi \leq \frac{H}{\lg k+\lg e}$. Further, if $H \geq \lg k$ then, $\pi \leq 1-\frac{k}{2^H}$.*

Proof: Using the same approach as in the proof of theorem 5.4.4 we have the expected number of faults incurred while processing a phrase is at most $\pi\lg_{\frac{k}{1-\pi}}(v_n(l-1)+1)$ for the following reasons. At each node the probability of not taking any of the k largest subtrees is at most π by using Rissanen's interpretation. Also whenever we incur a page fault the branch we choose has at most $(1-\pi)/k$ fraction of leaves of the parent subtree. Thus the minimal expected fault rate π is bounded

above by

$$\pi \leq \frac{v_n}{n}\pi \lg_{\frac{k}{1-\pi}}(v_n(l-1)+1) \tag{5.7}$$

$$\leq \pi \frac{H}{\lg k - \lg(1-\pi)} \tag{5.8}$$

$$\leq \frac{H}{1/\pi \lg k - 1/\pi \lg(1-\pi)} \tag{5.9}$$

$$\leq \frac{H}{\lg k + \lg e} \tag{5.10}$$

Further since $\pi \geq 0$ we have from equation 5.8, $\lg k - lg(1-\pi) \leq H$ which gives $\pi \leq 1 - \frac{k}{2^H}$, provided $k \leq 2^H$. □

5.5 Caching

In this section we study online *caching* or *demand paging*, where a page is fetched into cache only when a page fault occurs. By comparing the fault rates of two request sequences with equal entropy we will show that entropy of the request sequence alone does not fully capture the performance of online caching algorithms. Our construction uses the following two facts:

A prefetching algorithm can "simulate" a caching algorithm by prefetching at each step the k elements that are in the cache of the caching algorithm at that step. Thus, a lower bound on the fault rate of any prefetching algorithm for a given request sequence is also a lower bound on the fault rate of any caching algorithm on that sequence.

Consider a request sequence generated by a discrete memoryless source. It can be shown that the optimal online algorithm for caching in this case always keeps the $k-1$ pages with the highest probability in the cache, and leaves one slot for cache miss [33]. Thus, we can state the following theorem which follows from theorems 5.4.1 and 5.4.4. (Note that $L(H,k)$ and $U(H,k)$ are monotonically decreasing functions of k, assuming H and l are fixed.)

Theorem 5.5.1 *The best expected fault rate for any caching algorithm with cache size k on a request sequence generated by a discrete memoryless source with entropy H, is*

$$L(H,k) \leq \pi(k) \leq U(H,k-1).$$

Our construction uses request sequences generated by a Markov source.

Definition 5.5.1 *[34] A probabilistic finite state automaton (probabilistic FSA) as a quintuple $(S, \mathcal{H}, g, p, z_0)$ where S is a finite set of states with $|S| = s$, $\mathcal{H}$ is a finite alphabet of size l, g is a deterministic "next state" function that maps $SX\mathcal{H}$ into S, p_z is a "probability assignment function" for each $z \in S$ that maps $\mathcal{H}$ into $[0,1]$ with the restriction that $\sum_{i \in \mathcal{H}} p_z(i) = 1$ and $z_0 \in S$ is the start state. A probabilistic FSA when used to generate strings is called a* Markov source. *A Markov source is* ergodic *if it is irreducible and aperiodic, meaning that each state can reach every other state, and the gcd of the possible recurrence times for each state is 1. A Markov source is*

stationary when the start state is chosen randomly according to the steady state probabilities of the states.

A Markov source is a very general model and is not to be confused with a Markov chain on the page request sequence which is of first order. A Markov source can have infinite order. A stationary ergodic process can be approximated by a kth order Markov process, for large k [23]. We can define the entropy of a stationary Markov source as follows.

Definition 5.5.2 *[34] The entropy of a Markov source M denoted by $(S, \mathcal{H}, g, p, z_0)$ is given by*

$$H_M = \sum_{z=1}^{s} q(z)H(z)$$

where $q(z)$ is the stationary (steady state) probability corresponding to state z and $H(z)$ is the entropy of the state z defined as $-\sum_{x\in\mathcal{H}} p_z(x) \lg p_z(x)$.

Consider a two state Markov source with the same probability assignment function $p(.)$ for both states. Let H be the entropy of $p(.)$. Then the entropy of the Markov source is also H. We consider two cases:

Case 1 The pages corresponding to the top $k-1$ probabilities are the same in both states. In this case the best caching strategy is similar to the discrete memoryless case, that is keep the $k-1$ pages always in the cache. Hence, the fault rate $\pi(k)$ has the same bounds as in theorem 5.5.1.

Case 2 The set of $k-1$ pages with the highest probabilities in state 1 is disjoint from the set of $k-1$ pages with the highest probabilities in state 2. Suppose the stationary probabilities of the two states are $1/2$ each and the transition probability from each state to the other is also $1/2$. Then it can be shown that the best caching algorithm is to keep the top $(k-1)/2$ pages of each state (assuming k is odd) in the cache. Hence the minimal expected fault rate is (by theorems 5.4.1 and 5.4.4) in the range:

$$L(H, k/2) \leq \pi(k) \leq U(H, (k-1)/2)$$

It can be shown that the intervals corresponding in the above two cases are disjoint if k is sufficiently large. Thus, although the entropy in the two scenarios are equal, the fault rates are different.

5.6 Discussion

We briefly discuss how our approach may be used in certain situations to allocate resources more effectively. Consider the situation where we need to partition a (malleable) cache (for prefetching) for different data sources. One plausible way is to allocate more space in the cache for a data stream with lower entropy (assuming a common source alphabet). The motivation for this comes from our bounds on the fault rate for prefetching based on entropy of the source. Our bounds show that

the fault rate grows linear in H. From our lower bound (theorem 5.4.1) we get $k \geq 2^{\frac{H-1-\pi lgl}{1-\pi}}$. If we fix π our bound tells us that we need a larger cache for a data stream with higher entropy to achieve the same fault rate as opposed to a stream with lower entropy. Thus our bounds can be used in practice to partition a malleable cache in a better way according to the desired fault rates for different streams.

A very interesting open question which arises from our work is how accurately can entropy characterize performance of caching or list accessing algorithms in our model where requests are generated by a stationary ergodic source. Can we give an upper bound on the average work done by any list accessing algorithm as a function of entropy alone? We feel that entropy *alone* does not characterize the performance of any (or the best) online list accessing algorithm. We know that prefetching gives a (weak) lower bound on the minimal expected fault rate for caching (demand paging). Here also it seems that entropy alone does not tightly capture the minimal expected fault rate. It would be interesting to come up with a parameter which along with entropy will determine or (at least tightly upper bound) the performance of a caching algorithm. Another interesting area of research is to explore whether entropy gives good performance bounds for other online problems known in literature.

Chapter 6

Web Models

In this chapter, we develop and analyze new stochastic models for the Web graph which capture a global property of the Web: the *PageRank* distribution. We study the distribution of PageRank values (used in the Google search engine) on the Web. We show that PageRank values on the Web follow a power law. We then develop detailed models for the Web graph that explain this observation, and moreover remain faithful to previously studied degree distributions (a "local" property of the Web). We analyze these models, and compare the analysis to both snapshots from the Web and to graphs generated by simulations on the new models. To our knowledge this represents the first modeling of the Web that goes beyond fitting degree distributions on the Web.

6.1 Introduction

There has been considerable recent work on developing increasingly sophisticated models of the structure of the Web [1, 9, 10, 11, 18, 51, 52]. The primary drivers for such modeling include developing an understanding of the evolution of the Web, better tools for optimizing Web-scale algorithms, mining communities and other structures on the Web, and studying the behavior of content creators on the Web. Prior modeling has dwelt on fitting models to the observed degree distribution of the Web. While this represents a significant step (both empirically and analytically), a troubling aspect of this approach is the heavy reliance on a single set of parameters – the degree distribution. Moreover, the degree distribution is a very "local" property of graphs, something that is well recognized from at least two distinct viewpoints: (1) as a ranking mechanism, ordering the Web pages in search results by in-degree (popularity of linkage) is very easy to spam and thus not reliable; (2) from a graph-theoretic standpoint, it is easy to exhibit "very different" graphs that conform to the same degree distribution. Indeed, the first of these reasons led to the PageRank function [17] used in the Google engine.

In this chapter we present a more detailed approach to modeling, to explain the distributions of *PageRank* values on the Web. Our model augments the degree distribution approach, so that as a by-product we achieve previous models' success in explaining degree distributions.

There is a second, independently interesting set of reasons for our study of PageRank distributions. For search engines employing PageRank and associated ranking schemes, it is important to understand whether, for instance, 99% of the total PageRank is concentrated in (say) 10% of the pages. This (especially in conjunction with query distribution logs) has implications for compressing inverted indices and optimizing the available storage. A related question that arises: is PageRank strongly correlated with in-degree? (Most commonly, non-technical explanations of PageRank take the form "like the in-degree, except it matters where the pointers come from".) Beyond the issue of folklore versus reality, there is a substantial technical question here: could it be that PageRank is highly correlated to (say) in-degree, and thus the computational overhead (and ranking magic) of PageRank boils down to a simple popularity count by in-degree? Clearly one can concoct graphs for which the PageRank and degree distributions are highly correlated, just as one can concoct graphs for which they are not – but what happens on the true Web?

We develop a series of experiments to resolve these questions. In the process we develop more detailed models of Web graph evolution than in prior work, and demonstrate on simulations as well as on extracts of the Web that our new model better fits the empirical evidence.

6.2 Background and Main Contributions

We begin by reviewing related background in Section 6.2.1; the reader familiar with this material may wish to skip ahead to Section 6.2.2.

6.2.1 Preliminaries

We now set the stage for discussing graph models of the Web, beginning with the standard view of the Web as a graph (Section 6.2.1). We next review the basics of the PageRank function [17] reportedly used in the Google search engine (Section 6.2.1).

The Web as a Graph

View the Web as a *directed* graph whose nodes are html pages. Each hyperlink is a directed edge in the natural manner. The *in-degree* of a node is the number of edges (hyperlinks) into it; a simplistic interpretation of the in-degree of a page is as a popularity count. The *out-degree* of a node is the number of links out of it; this is simply the number of `href` tags on the page. The *degree distribution* of a graph is the function of the non-negative integers that specifies, for each $k \geq 0$, what fraction of the pages have degree k; there are naturally two degree distributions for a directed graph, the in-degree distribution and the out-degree distribution.

These distributions have been the objective of considerable prior study [1, 9, 10, 11, 18, 51, 52], on various snapshots of the Web ranging from the Web pages at a particular university to various commercial crawls of the Web. Despite the varying natures of these studies, the in-degree distribution appears to be very well approximated by the function $c/k^{2.1}$ where c is the appropriate normalization

constant (so that the fractions add to one). Likewise, the out-degree distributions seem to be very well approximated by the function $c_o/k^{2.7}$. Such distributions are known as *power law* distributions.

Recent work of Dill et al. [26] provides some explanation for this "self-similar" behavior: that many properties of the Web graph are reflected in sub-domains and other smaller snapshots of the Web. Indeed, this will provide the basis for some of our experiments, in which we derive an understanding of certain properties of the Web by studying a crawl of the `brown.edu` domain. (This methodology was pioneered by Barabasi et al. [9, 10, 11], who extrapolated from the `nd.edu` domain of Notre Dame University. They made a prediction on the diameter of the undirected version of the Web graph, in which one ignores link directions.)

Other properties of the Web graph that have been studied (analytically or empirically) include connectivity [18], clique distributions [51] and diameter [15].

PageRank Primer

The *PageRank* function was presented in [17, 59] and is reportedly used as a ranking mechanism in the commercial search engine Google [37]. It assigns to each Web page a positive real value called its PageRank. In the simplest use of the PageRank values, the documents matching a search query are presented in decreasing order of PageRank. We now briefly discuss the notion of PageRank and its practical implementation via the *decay* parameter.

The original intuition underlying PageRank was to visualize a random surfer who browsed the Web from page to page. Given the current location (page) q of the surfer, the successor location is a page reached by following a hyperlink out of page q uniformly at random. Thus each hyperlink is followed with probability proportional to the out-degree of q. In this setting, the PageRank of each page is the frequency with which, in the steady state, the page q is visited by such a surfer. Intuitively, the surfer frequently visits "important" pages such as `yahoo.com` because many pages hyperlink to it. Moreover, by calculations from elementary probability theory, the PageRank of a page q is increased if those pages that hyperlink to q have high PageRank themselves.

An immediate difficulty with this notion: some pages, or an (internally) connected cluster of pages may have no hyperlinks out of them[1], so that the random surfer may get stuck. To address this, Brin and Page [17] introduced the following device: at each step, with some probability, the surfer "teleports" to a completely random Web page, independent of the hyperlinks out of the current page. At least in consideration of the surfing behavior of early users of the Web (from the mid 1990's), such serendipitous teleporting followed by some depth-first exploration (before teleporting again) was reasonable. More important to the notion of PageRank, it removes the technical difficulty created by (connected clusters of) pages having no hyperlinks out of them.

Let the pages on the Web be denoted by $1, 2 \ldots, m$. Let $d_{out}(i)$ denote the number of outgoing links from page i, i.e., the out-degree of i. Let $In(i)$ denote the set of pages that point to i. Let $p (0 < p < 1)$ be the *decay factor* that represents the probability with which the surfer proceeds with

[1] Content from certain disciplines – such as the humanities – tends to be "less hyperlinked" and more in the form of monologues and discourse without links than disciplines close to computing and the Web [35].

the random walk, while $1-p$ is the probability of teleporting to a random page amongst all m Web pages. Then the PageRank $r(i)$ of page i is given by ([17]):

$$r(i) = \frac{1-p}{m} + p * \sum_{j \in In(i)} \frac{r(j)}{d_{out}(j)}.$$

This represents a system of linear equations (one for each $i \in \{1, 2, \ldots, m\}$). We may rewrite this in matrix form, and the unique solution vector $r(i)$ can be expressed as the eigenvector of a matrix [17, 59] or as the stationary probability of a random walk [56] (thus $\sum_i r(i) = 1$). Details are beyond the scope of this brief exposition.

While we will not get deeper into the mathematical underpinnings of PageRank here, it should be intuitively clear that the PageRank values of pages are global properties (in contrast to the more local nature of in-degree). One could in principle concoct examples in which the PageRanks of a few nodes could be "engineered", but fitting the entire distribution is much harder. This observation is one reason why we propose that the PageRank distribution is a far more reliable characteristic to model, than the degree distribution. Moreover, as we show below, our model captures the PageRank distribution while remaining faithful to the degree distribution.

6.2.2 Main contributions and Organization of the Chapter

We review graph models in Section 6.3. We augment the current set of models by proposing a new model – which we call *PageRank-based selection* – in which attachment probabilities for new hyperlinks are based on the PageRanks of existing nodes. The intent in proposing this model is to explain our empirical observations on PageRank distributions, described below. We suggest a behavioral explanation of content creation that might underlie this model. We also present a *hybrid selection model* that is a natural combination of previous models with our PageRank-based selection model.

In Section 6.4 we describe experiments on snapshots from the Brown University Web, as well as from the publicly available WT10g Web snapshot. Our first finding is that the PageRank distribution follows a power law with exponent 2.1. This is extremely interesting for several reasons: (1) PageRank is distributed as a power law; (2) it has the same exponent (namely, 2.1) as that observed for in-degree on many independent snapshots of the Web; (3) the distribution is (as already known for in- and out-degree distributions) relatively insensitive to the particular snapshot of the Web on which the measurement is made. The fact that in-degree and PageRank follow similar power law distribution on the Web graph might lead to the conclusion that the two properties are highly correlated. This, however, is not the case for the Web graph. Our experiments show very little correlation between the two properties on the Web graph. A high in-degree of a node does not imply high PageRank and vice versa.

Section 6.5 adopts analytical as well as simulation-based approaches to validating our models and fitting model parameters. We first present analysis based on the "mean-field" approach [9, 10, 11] that the classical degree-based selection model as well as our new PageRank-based selection yield

power laws for the PageRank distribution. The question then is whether the exponents predicted by the analysis match the observations. Given that these are parameterized models, we are able to find combinations of models and parameters that do indeed fit both the PageRank and degree distributions. We verify that these models do generate graphs with the correct distributions through simulations in which we generate multiple random graphs and measure their distributional properties (Section 6.5.3).

To our knowledge, these are the first results that capture global distributional properties in a model, validating empirical observations through analysis and simulation. Our new models simultaneously capture degree distributions – local properties studied in previous models. We suggest behavioral explanations for our models, allowing the prediction of what would happen to PageRank and degree distributions if more content creators were to link to pages ranked highly by PageRank-based search engines such as Google.

6.3 Web Graph Models

In the *Erdös-Renyi* model of random graphs [13], each edge is directed from a node to another node that is chosen uniformly at random from all the other nodes in the graph. There is a wealth of research on such graphs, and many properties of such random graphs are well understood. For instance, an Erdös-Renyi random graph in which the average out-degree of each node is roughly 7 (as is the observed average out-degree of Web pages), the degree distributions are Poisson, and it is extremely unlikely that there are any clique-like structures with more than a handful of nodes. Given the many consistent observations of power law degrees on the Web graph, as well as the superabundance of clique-like structures [52], it is clear that the Web graph does not conform to the Erdös-Renyi model. Nevertheless, as we will see below, elements of random selection do play a role in models that are more faithful to the Web graph.

A number of research projects proceeded to develop models that better explained the power law behavior of degree distributions on the Web; see [60] for a survey of these. In all of these, the view is that of nodes and edges being added to the graph one at a time. As noted above, it does not suffice for such newly arriving edges to choose to point to a node (page) chosen uniformly at random, since this does not yield a power law distribution for degrees. The simplest model to overcome this problem uses the following device: each edge chooses the node to point to at random, but with non-uniform probabilities for choosing the various nodes. In particular, the edge points to a node q in proportion to the current in-degree of q. This yields Web graphs whose in-degree distributions have been shown to converge to the distribution $\approx 1/k^2$ [9, 10, 11].

However, as noted earlier, empirical studies have shown that in-degrees are in fact distributed as $\approx 1/k^{2.1}$ (rather than $1/k^2$). To help explain the exponent of 2.1, Kumar *et al.* [53] introduced the following more detailed process by which each edge chooses the node to point to. Some fraction of the time (a parameter they call $\alpha \in [0,1]$) the edge points to a node chosen uniformly at random. The rest of the time (a fraction $1-\alpha$), the edge picks an intermediate node v at random, and *copies*

the destination of a random edge out of v. In other words, the new edge points to the destination of an edge e, chosen at random from the outgoing edges of a random node v. Kumar *et al.* offer the following behavioral explanation for this process: some fraction of the time a content creator creating a page refers to a random new topic and thus creates a link (edge) to a random destination. The remainder of the time, the content creator copies a hyperlink off an existing page (in this case v), having decided that this is an interesting link. They then explain a number of empirical observations on the Web graph including the in-degree exponent of 2.1 and the large number of clique-like structures observed by [52]. In fact, they prove theorems that derive the exponent as a function of the parameter α. There is another way of viewing this model: a fraction α of the edges go to random nodes, while the remainder choose destination nodes in proportion to their current degrees. Thus, their model may be viewed as a generalization of the models of Barabasi and others, parameterized by α. We will henceforth refer to this model as the *degree-based selection model*. Could it be that this model would also explain the PageRank distributions we observe on the Web?

Before we address this question, we next introduce a new model inspired by the α model above. Suppose that each edge chose its destination at random a fraction $\beta \in [0,1]$ of the time, and the rest of the time chose a destination in proportion to its *PageRank*. Following the behavioral motivation of Kumar *et al.*, this can be thought of as a content-creator who chooses to link to random pages some fraction of the time, and to pages highly rated by a PageRank-based engine such as Google the remainder of the time. In other words, content creators are more likely to link to pages that score high on PageRank-based search results, because these pages are easy to discover and link to. This is not implausible from the behavioral standpoint, and could help capture the PageRank distributions we observe (just as in-degree based linking helped explain in-degree distributions in prior work). We will call this the *PageRank-based selection model*.

However, this now raises the question: if we could develop a model that explained observed PageRank distributions, could it be that we lose the ability to capture observed degree distributions? To address this, we now present the most general model we will study. There are two parameters $a, b \in [0,1]$ such that $a + b \leq 1$. With probability a an edge points to a page in proportion to its in-degree. With probability b it points to a page in proportion to its PageRank. With the remaining probability $1 - a - b$, it points to a page chosen uniformly at random from all pages. We thus have a family of models; using these 2-parameter models we can hope to simultaneously capture the two distributions we investigate – the PageRank distribution (representing global properties of the graph), and the in-degree distribution (representing local properties of the graph). We will call this the *hybrid selection model*.

6.4 Experiments

To set the context for exploring the models in Section 6.3, we study the distribution of PageRanks (as well as of the in- and out-degrees) on several snapshots of the Web.

6.4.1 Experiments on the Brown University Domain

Our first set of experiments was on the Web graph underlying the Brown University domain (`*.brown.edu`). Our approach is motivated by recent results on the "self-similar" nature of the Web (e.g., [26]): a thematically unified region (like a large subdomain) displays the same characteristics as the Web at large. The Brown Web consisted of a little over 100,000 pages (and nearly 700,000 hyperlinks) with an average in-degree (and thus out-degree) of around 7. This is very close to the average in-degree reported in large crawls of the Web [52]. Our crawl started at the Brown University homepage (`www.brown.edu` – "root" page) and proceeded in breadth-first fashion; any URL outside the `*.brown.edu` domain was ignored. We did prune our crawl – for example, URL's with /cgi-bin/ were not explored.

The graphs shown in Figures 6.1, 6.2 and 6.3 summarize our results on the in-degree, out-degree and the PageRank distributions in the Brown Web graph[2]. Our experiments show that the in-degree and out-degree distribution follows a power law with exponent 2.1 and 2.7 respectively. This is strikingly similar to the results reported on far larger crawls of the Web [18, 52]. For example [18] report exactly the same power law exponents on a crawl of over 200 million pages and 1.5 billion hyperlinks.

However, the most interesting result of our study was that of the PageRank distribution. We first describe our PageRank computation. As in [59], we first pre-process pages which do not have any hyperlinks out of them (i.e., pages with out-degree 0): we assume that these have links back to the pages that point to them [2]. This is intuitively more justifiable than just dropping these pages: we expect surfers to trace back their trail when they reach a dead end. In our PageRank computation we set the decay parameter to 0.9; this is a typical value reportedly used in practice (e.g., [17] uses 0.85), and the convergence is fast (under 20 iterations). Similar fast convergence is reported in [59, 17]. However, varying the decay parameter does not significantly change our results, as long as the parameter is fairly close to 1. In particular, we get essentially the same results for decay parameter values down to 0.8.

The main result of our PageRank distribution plot is that a large majority of pages (except those with very small PageRank) follow power law with an exponent close to 2.1. That is, the fraction of nodes having PageRank r is proportional to $1/r^{2.1}$. This appears to be the same as the in-degree exponent; more on this later. In section 6.5 we will give an analysis suggesting this PageRank distribution, based on various models from Section 6.3.

We also note that the distribution is almost flat for pages with very low PageRank. To check whether this is an anomaly, we repeated the experiments for the Brown Computer Science department subdomain (`*.cs.brown.edu`) and we got almost identical results (i.e., in-degree, out-degree and PageRank distributions follow power laws with almost identical exponents) even though `*.cs.brown.edu` is a much smaller graph (around 25,000 nodes); and a similar flattening at the top

[2] To avoid excessively "dark" plots resulting in large amounts of redundant data, all plots in this chapter have been sub-sampled.

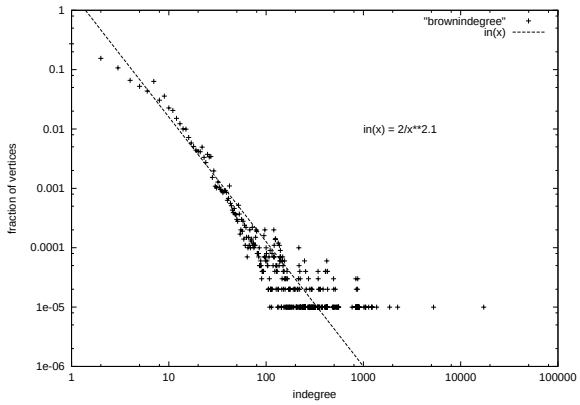

Figure 6.1: Log-log plot of the in-degree distribution of the Brown domain (`*.brown.edu`). The in-degree distribution follows a power law with exponent close to 2.1.

(corresponding to pages with very low PageRank) in the PageRank distribution. When comparing this pattern to the experiments on the WT10g corpus (next subsection) that captures a more generic subset of the Web, we conclude that relatively structured domains, such as `brown.edu` and `cs.brown.edu`, have a smaller fraction of very "unimportant" pages than predicted by the power law distribution and observed in less structured corpora.

To understand the correlation between in-degree (and out-degree) and PageRank we scatter-plotted the PageRank verses the in-degree and out-degree. These are shown in Figures 6.4 and 6.5 respectively. It clearly shows that there is very little correlation between PageRank and in-degree, except possibly when the in-degree is very high. Thus PageRank is indeed a characteristic of pages that is not predicted by the in-degree. In particular, when ranking search results, "highly relevant pages" (those with high PageRank) could have low in-degree. The correlation coefficient between the PageRank and in-degree is 0.35 and between the PageRank and out-degree is 0.34. This is indeed surprising given the similarity of the slopes of the two distributions. We will return to this later.

6.4.2 Experiments on WT10g Data

We repeated our experiments on the WT10g corpus [80], a recently released, 1.69 million document testbed for conducting Web experiments. The results are almost identical to those on the Brown Web; the in-degree, out-degree, and PageRank distributions follow power laws with exponent close to 2.1, 2.7 and 2.1 respectively. Figure 6.6 shows the plot of PageRank distribution of the wt10g corpus (we are not showing the in-degree and out-degree distribution plots as they are very similar to those of the Brown Web). The power law here appears much sharper than in the Brown Web. Also, unlike the Brown Web, the plot has slope 2.1 almost the entire spectrum of PageRank values, except for those with very low PageRank values. As noted above, a possible explanation is that unlike the Brown domain, the WT10g corpus is constructed by a careful selection of Web pages so as to characterize the *whole Web* [80].

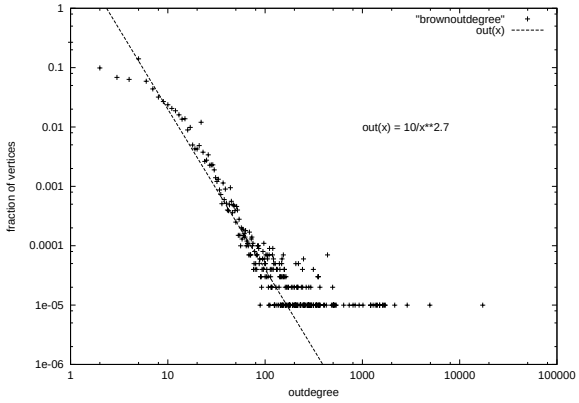

Figure 6.2: Log-log plot of the out-degree distribution of the Brown domain (`*.brown.edu`). The out-degree distribution follows a power law with exponent close to 2.7.

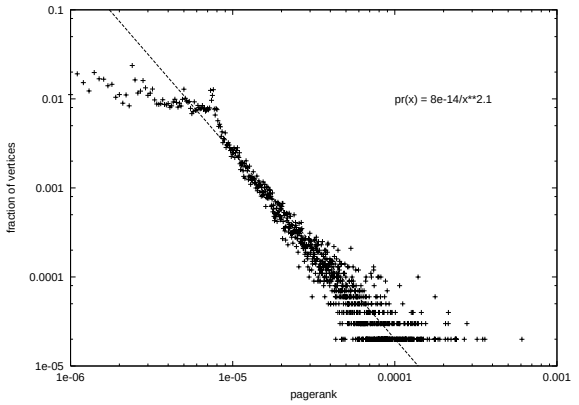

Figure 6.3: Log-log plot of the PageRank distribution of the Brown domain (`*.brown.edu`). A vast majority of the pages (except those with very low PageRank) follow a power law with exponent close to 2.1. The plot almost flattens out for pages with very low PageRank.

We also repeated our correlation experiments on this corpus: there is almost no correlation between the PageRank and in-degree (out-degree) distributions. The correlation coefficient is 0.15 (0.07). This is even less than the correlation observed on the Brown domain: again, this may be due to the fact that unlike the Brown domain, the corpus is more representative of the Web (and also much larger).

6.5 Fitting the Models: Analysis and Simulations

In this section we address some of the modeling questions raised in section 6.3. Having obtained the empirical distributions in Section 6.4, we first give analytical predictions of the shape of the PageRank distributions for the degree-based and PageRank-based selection models of Section 6.3.

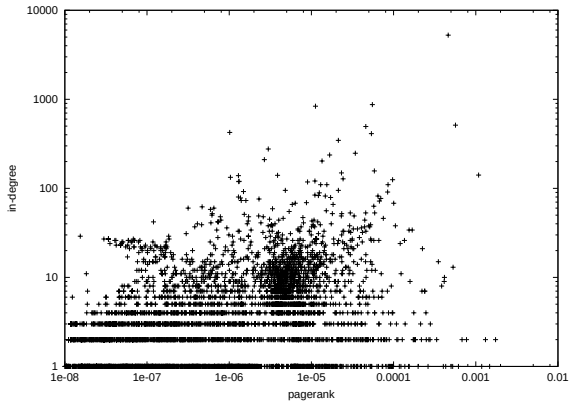

Figure 6.4: Log-log scatter plot of the PageRank verses the in-degree of the Brown domain, showing very little correlation. The corresponding graph for the WT10g corpus is very similar.

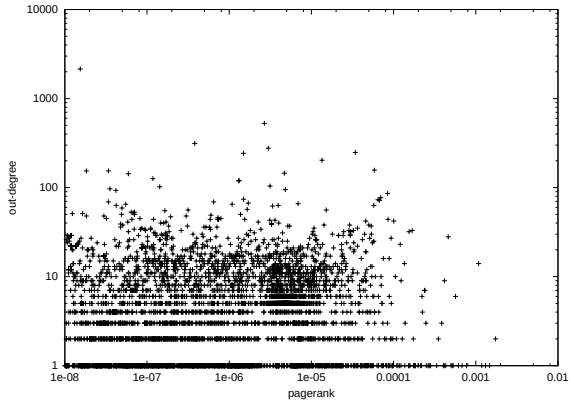

Figure 6.5: Log-log scatter plot of the PageRank verses the out-degree of the Brown domain, showing very little correlation. The corresponding graph for the WT10g corpus is very similar.

The intent is to infer what choices of these model parameters would give rise to the distributions observed in our experiments. Finally, in Section 6.5.3 we generate random graphs according to these fitted models, to see if in fact they give rise to graphs that match the distributions observed on the Web.

6.5.1 Degree-based Selection

Consider a graph evolving in a sequence of *time steps* – as noted in Section 6.3 such evolution is not only realistic in the context of the Web, it is also a feature of all Web graph models. A single node with r outgoing edges is added at every time step. (We assume that we start with a single node with a self-loop at time 0 [14]).) Each edge chooses its destination node independently with probability proportional to 1+in-degree[3] of each possible destination node. This model is essentially the one

[3]We assume that each incoming node has "weight" 1, otherwise there won't be any non-trivial growth.

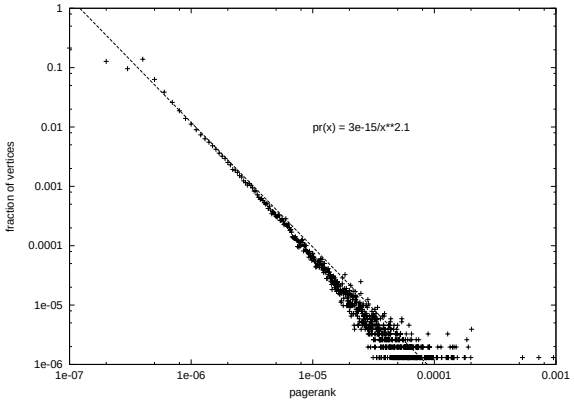

Figure 6.6: Log-log plot of the PageRank distribution of the WT10g corpus. The slope is close to 2.1. Note that the plot looks much sharper than the corresponding plot for the Brown Web. Also, the tapering at the top is much less pronounced.

analyzed by Barabasi *et al.* and is a special case of the α model of Kumar *et al.*

Let $\pi^t(v)$ represent the PageRank of v at time step t. We can interpret the PageRank as the stationary probability of a random walk on the underlying graph, with the teleport operation (Section 6.2.1) being modeled by a "central" node c. At each step, the surfer either decides to continue his random walk with probability p or chooses to return to the central node with probability $1-p$; from the central node he jumps to a random node in the graph. To write an expression for $\pi^t(v)$ it is useful to define $f^t(v)$, the "span" of v at time t: the *sum* of the in-degrees of all nodes in the network (including v itself) that have a path to v that does not use the central node (we also refer to the nodes contributing to the span as "span nodes"). Since each edge contributes a $1/r$ fraction of the stationary probability of its source node (using the standard stationary equations (see [56])), we can bound $\pi^t(v)$ for the above random walk as follows:

$$\frac{f^t(v)\pi(c)p^D}{rt} \leq \pi^t(v) \leq \frac{f^t(v)\pi(c)}{rt} \tag{6.1}$$

where $\pi(c)$ is the stationary probability of the central node and D is the diameter of the network (ignoring link directions). We note two facts here. First, a simple observation shows that $\pi(c)$ is a constant, independent of t; second, it can be shown that when t is sufficiently large, the diameter of the graph at time t is logarithmic in the size of the graph (which is t) [15]. Thus if the decay factor p is sufficiently close to 1, we can approximate $\pi^t(v)$ as

$$\pi^t(v) \approx \frac{f^t(v)\pi(c)}{rt}. \tag{6.2}$$

We now proceed to estimate $f^t(v)$. Following the "mean-field" approach of Barabasi *et al.* [11], and treating $f^t(v)$ as continuous, we can write the differential equation for the rate of change of $f^t(v)$ with time:

$$\frac{d(f^t(v))}{dt} = \frac{f^t(v)}{t} \tag{6.3}$$

where the right hand side denotes the probability that an incoming edge connects to one of the span nodes of v. The solution to (Equation 6.3) with the initial condition that node v was added at time t_v is

$$f^t(v) = \left(\frac{t}{t_v}\right). \tag{6.4}$$

Combining Equations (6.2) and (6.4), we have

$$\pi^t(v) \approx \frac{\pi(c)}{rt_v}. \tag{6.5}$$

Using the above equation,

$$\Pr(\pi^t(v) < \phi) = \Pr(t_v > \frac{\pi_c}{r\phi}).$$

Since nodes are added at equal time intervals, the probability density of t_v is $1/t$. Thus we obtain

$$\Pr(t_v > \frac{\pi_c}{r\phi}) = 1 - \Pr(t_v \leq \frac{\pi_c}{r\phi}) = 1 - \frac{\pi_c}{rt\phi}$$

which yields that the probability density function F for $\pi^t(v)$ is:

$$F(\phi) = \frac{\partial(\Pr(\pi^t(v) < \phi))}{\partial\phi} \approx \frac{\pi(c)}{rt\phi^2} \tag{6.6}$$

implying that the PageRank follows a power law with exponent 2, independent of r and t. Simulations of this model (described below and shown in Figure 6.7) agree well with this prediction.

As already mentioned in Section 6.3 the in-degree distribution of this model follows a power law with exponent 2, the same as the PageRank distribution derived above. This is striking given that in our empirical studies too, the in-degree and PageRank distributions had identical power laws. However, the empirically observed power laws have exponents of 2.1; thus the degree-based selection model does not quite match the in-degree and PageRank exponents observed in practice. Now a natural question is whether we can make it match both the distributions by changing α, i.e., by incorporating a random selection component in choosing nodes. The answer is surprisingly[4] yes; more on this in Section 6.5.3 below. But first we analyze PageRank-based selection.

6.5.2 PageRank-based Selection

We show that power law emerges for the PageRank and degree distributions in this model, but the exponents are different from the degree-based model.

Using the same argument as before, we can show that Equation (6.2) holds. However, $f^t(v)$ follows a different differential equation from Equation (6.3). Instead we have

$$\frac{d(f^t(v))}{dt} \approx \frac{f^t(v)r}{2rt}. \tag{6.7}$$

[4] Surprising because, it is not the case that PageRank and in-degree distributions are related, as our analysis might lead us to believe. Consider the uniform selection model. It can be shown by similar analysis that a power law (with a small exponent) emerges for the PageRank even here; but as mentioned in section 6.3 the degree distribution follows a Poisson distribution.

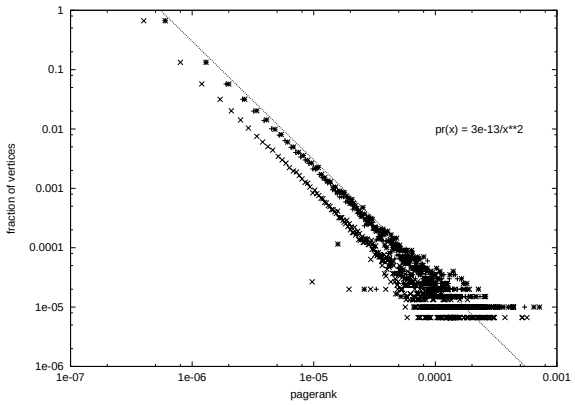

Figure 6.7: Log-log plot of degree-based selection with $\alpha = 0$. The number of nodes shown is 300,000 (+), 200,000 (*) and 100,000 (x). It clearly shows that the slope is 2, confirming the power law predicted by analysis.

The reasoning is as follows. The probability that $f^t(v)$ increases by one is the probability that the incoming node chooses any one of the nodes in the span to connect to, which is proportional to the sum of the PageRanks of all the span nodes of v. To calculate this probability, we see that each directed edge contributes nearly *twice* to the sum (if p is sufficiently large) and the total PageRank is thus proportional to the sum of the degrees which is $2rt$.

Plugging the solution of the above differential equation in Equation (6.2), we can show that the probability density function F for $\pi^t(v)$ in this model is:

$$F(\phi) \approx \frac{(\pi(c))^2}{r^2 t^2 \phi^3} \tag{6.8}$$

i.e., predicting that the PageRank follows a power law with exponent 3. Analogously, we can show that the degree also a follows a power law with exponent 3. Simulations also agree quite well with this prediction.

Thus, the PageRank-based selection model with $\beta = 0$ does not match the empirically observed in-degree and PageRank exponents. Can we hope to match the observations by varying β? Unlike the degree-based selection model, the answer is no; increasing β will only increase the power law exponent (above 3) for the in-degree distribution. This can be verified by experiments, as well as by a direct extension to the analysis above; details are omitted in this abstract. We are thus left with the degree-based selection model and the hybrid selection model of Section 6.3 as candidates for explaining the observations.

6.5.3 Simulations of the Generative Models

An accurate model of the Web graph must conform with the experimentally observed in-degree, out-degree, and PageRank distributions. We simulated the degree-based and hybrid selection models defined in section 6.3 under various parameters to find settings that generate the observed empirical

distributions. We simulated graphs of size up to 300,000 nodes, and we varied the average number of new edges generated per new node generation (time step). In particular, to be "close" to the real Web's average out-degree (and in-degree), we focused on the range in which the average number of edges added per new node is around 7. We obtained essentially the same results for the power laws, irrespective of the size (from 10,000 nodes onwards) or the number of outgoing edges.

Our first step was fitting the out-degree distribution. Following Kumar *et al.*, we use the degree-based copying model with a suitable value of β to fit the out-degree distribution to a power law with exponent 2.7. At each time step, the incoming node receives edges from existing nodes. With probability β a node is chosen uniformly at random, with probability $1 - \beta$ the node is chosen proportional to the current out-degree distribution. Note that the out-degree distribution is fixed independently of the in-degree distribution. We use $\beta = 0.45$ to get a power law exponent equal to 2.7.

We turn now to the problem of fitting the in-degree distribution. We first simulated the degree-based selection model. Setting $\alpha = 0$, both the in-degree and PageRank distributions followed a power law with exponent 2. We observed that increasing α increases the exponents in the in-degree and PageRank distributions. In particular, setting $\alpha \approx 0.2$ brings both exponents to the empirical value of 2.1. This value is unique; by increasing or decreasing α we lose the fit. Thus, we found a setting of the parameters for which the degree-based selection model simultaneously fits all the three distributions.

Since degree-based selection model fits the empirical data, a natural question is whether PageRank-based selection is irrelevant in modeling the Web graph. To answer this, we experimented with the 2-parameter hybrid selection model proposed in Section 6.3. Surprisingly when $a = b \approx 0.33$, we could again simultaneously fit all three distributions. Thus we have an alternative model, with a substantial PageRank-based selection component, that fits the Web empirical data. As mentioned in Section 6.3, this model is plausible from the behavioral standpoint.

To further understand these models we scatter-plotted the PageRank and in-degree distributions for the above two models: we found a very high correlation (close to 0.99) between PageRank and in-degree in both models, unlike the empirical Web data. We outline a possible explanation in the concluding section.

6.6 Conclusion and Further work

We present experimental and analytical study of PageRank distribution on the Web graph, and use it to develop more accurate generative models for the evolution of the Web graph. Our first finding is that PageRank distribution on snapshots of the Web graph follows a power law distribution with the same exponent as the in-degree distribution. Despite this similarity in distributions, our experiments show that there is very little correlation on the Web graph between a node's in-degree and PageRank. Thus, PageRank distribution is an independent feature of the Web graph. Furthermore, unlike in-degree, PageRank is a global property of the graph, thus one expects to obtain more accurate

modeling of the Web graph by fitting the models to the PageRank distribution.

We consider three possible models for the Web graph: degree-based selection model, PageRank-based selection model, and a hybrid model. Our analysis shows that the PageRank-based selection model cannot fit the empirical data. For the two other models we found settings of parameters under which the model fits simultaneously the in-degree and out-degree distributions and the PageRank distribution. A natural question for further study is whether one of these models describes the Web better than the other.

A second challenging question is extending these simple models to capture the important notion of communities and sub-communities on the Web. All models proposed and analyzed so far grow by making "global" choices: connections are chosen by various distributions, but from *all* the existing nodes. In practice, links between nodes cannot be fully explained just by the relative popularity of the nodes. While nodes are likely to link to important or popular nodes, these nodes are also likely to be in the same sub-community. We are exploring generative models that capture this feature, and expect these models to explain, among other things, the discrepancy between PageRank and in-degree correlations in the empirical data and the simulated models.

Epilogue

In this thesis we presented results in four different settings in which stochastic models proved useful in modeling dynamic computer phenomena. They were useful in two ways: providing mathematical insight into the essential nature of the phenomena as it occurs in real-life and in giving a reasonably realistic framework for designing and analyzing algorithms/protocols for associated problems. Thus one of the main goals of this thesis is to illustrate the usefulness of stochastic modeling and analysis in a variety of application areas.

Stochastic analysis does not fall under the category of worst-case analysis. In many scenarios worst-case analysis is meaningless or uninteresting or does not capture typical real-life situations. We can illustrate with a couple of examples. Competitive analysis is meaningless to analyze prefetching since the optimal offline algorithm will always prefetch correctly (Chapter 5). Worst-case analysis of our P2P protocol (Chapter 4) is both uninteresting and unrealistic: for instance, a malicious adversary can make sure that the network is disconnected at any point of time irrespective of what the protocol does. It also doesn't reflect what typically happens in real-life. On the other hand, our stochastic model is a more realistic approximation of real-life P2P systems as supported by experimental data.

Our preceding discussion leads to an important question: how to find a good mathematical model for the phenomenon? This is beyond the scope of this thesis, but we state two guiding principles which we used: a good model is one which reasonably approximates real-life data and/or captures an important feature of the phenomenon, and is *mathematically tractable*.

We hope our thesis will inspire further work in similar spirit: understanding dynamic computer phenomena that arise in modern applications and in designing and analyzing algorithms for associated problems using stochastic modeling and analysis.

Bibliography

[1] L. Adamic and B. Huberman. Power Law distribution of the World Wide Web, Technical Comment on [9], *Science*, **287**, 2000, 2115a.

[2] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, Sriram Raghavan. Searching the Web. *ACM Transactions on Internet Technology*, **1**(1), 2001, 2-43.

[3] N. Alon and J. Spencer, *The Probabilistic Method*, John Wiley, 1992.

[4] D. Anick, D.Mitra and M.M. Sondhi, "Stochastic Theory of a Data Handling System with Multiple Sources". *Bell Systems Technical Journal*, **61**(8), 1982, 1871-1894.

[5] S. Albers and M. Mitzenmacher, Average Case Analysis of List Update Algorithms, with Applications to Data Compression, *Algorithmica*, **21**, 1998, 312-329.

[6] P. Algoet, Universal Schemes for Prediction, Gambling and Portfolio Selection, *Annals of Probability*, **20**(2), 1992, 901-941.

[7] P. Algoet and T.M. Cover, Asymptotic Optimality and Asymptotic Equipartition Property of Log-Optimal Investment, *Annals of Probability*, **16**, 1988, 876-898.

[8] K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, **19**, 357-367, 1967.

[9] A. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science* , **286**(509), 1999.

[10] A. Barabasi, R. Albert, and H. Jeong. Scale-free characteristics of random networks: The topology of the World Wide Web. *Physica A*, **281**, 2000, 69-77.

[11] A. Barabasi, R.Albert and H. Jeong. Mean-field theory for scale-free random graphs. *Physica A*, **272**, 1999, 173-187.

[12] J.L. Bentley, D.D. Sleator, R. E. Tarjan and V.K. Wei, A Locally Adaptive Data Compression Scheme, *Communications of the ACM*, **29**(4), 1986, 320-330.

[13] B. Bollobas, *Random Graphs*, Academic Press, 1985.

[14] B. Bollobas, O. Riordan, J. Spencer, and G. Tusnady. The degree sequence of a scale-free random graph process. *Random Structures and Algorithms*, **18**(3), 2001, 279-290.

[15] B.Bollobas and O. Riordan. The diameter of a scale-free random graph. *preprint*, 2001.

[16] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, 1998.

[17] S. Brin and L. Page. The anatomy of a large-scale hypertexual Web search engine. In *Proceedings of the 7th WWW conference*, 1998.

[18] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, Andrew Tomkins, J. Weiner. Graph Structure in the Web. In *Proceedings of the 9th WWW Conference*, 2000.

[19] Clip2, "Gnutella Measurement Project", May 2001. *http://www.clip2.com*

[20] D. Clark, Face-to-Face with Peer-to-Peer Networking, *Computer*, **34**(1), 2001.

[21] I. Clarke. A Distributed Decentralized Information Storage and Retrieval System, Unpublished report, Division of Informatics, University of Edinburgh (1999).

[22] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, Freenet: A distributed anonymous information storage and retrieval system, In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, 2000. (http://freenet.sourceforge.net)

[23] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[24] K. Curewitz, P. Krishnan and J.S. Vitter, Practical Prefetching Via Data Compression, In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1993, 257-266.

[25] Derek Chiou, Prabhat Jain, Srinivas Devadas and Larry Rudolph, Dynamic Cache Partitioning via Columnization, in *Proceedings of Design Automation Conference*, Los Angeles, June 2000.

[26] S. Dill, R. Kumar, K. McCurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-Similarity in the Web. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB)*, 2001.

[27] P. Elias, Universal Codeword Sets and the Representation of the Integers, *IEEE Transactions on Information Theory*, **21**(2), 1975, 194-203.

[28] A.Elwalid and D.Mitra. Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks. *IEEE/ACM Trans. Networking*, **1**(3), 1993, 329-343.

[29] A. Fiat, R.M. Karp, M. Luby, L. A. McGeoch, D.D. Sleator and N.E. Young, On Competitive Algorithms for Paging Problems, *Journal of Algorithms*, **12**, 1991, 685-699.

[30] M. Feder and N. Merhav, Relations between Entropy and Error Probability, *IEEE Transactions on Information Theory*, **40**(1), 1994, 259-266.

[31] M. Feder, N. Merhav and M. Gutman, Universal Prediction of Individual Sequences, *IEEE Transactions on Information Theory*, **38**, 1992, 1258-1270.

[32] W. Feller, *An Introduction to Probability Theory and its Applications*, volume I, John Wiley, New York, 1968.

[33] P.A. Franaszek and T.J. Wagner. Some Distribution-free Aspects of Paging Performance, *Journal of the ACM*, **21**, 1974, 31-39.

[34] R.G. Gallager, *Information Theory and Reliable Communication*, Wiley, New York, 1968.

[35] D. Gibson, J.M. Kleinberg and P. Raghavan. Inferring Web communities from link topology. In *Proceedings of the ACM Symposium on Hypertext and Hypermedia*, 1998.

[36] G.H. Gonnet, J.I. Munro, and H. Suwanda. Exegesis of Self-organizing Linear Search, *SIAM Journal of Computing*, **10**, 1982, 613-637.

[37] Google Inc. `http://www.google.com`

[38] S. Gupta, K. Ross and M. Zarki. On Routing in ATM Networks. In [73].

[39] The Gnutella Protocol Specification v0.4. *http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf*

[40] Gnutella website. *http://gnutella.wego.com/*.

[41] M.E. Hellman and J. Raviv, Probability of Error, Equivocation and the Chernoff Bound, *IEEE Transactions on Information Theory*, **16**(4), 1970, 368-372.

[42] J.L. Hennessey and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd edition, Morgan Kaufmann, 1996.

[43] J.Y. Hui. Resource Allocation for Broadband Networks. *IEEE J. Selected Areas in Comm.*, **6**, 1988, 1598-1608.

[44] A.R. Karlin, S.J. Phillips and P. Raghavan, Markov Paging, In *Proc. of the 33rd IEEE Symposium on the Foundations of Computer Science (FOCS)*, 1992, 208-217.

[45] S. Karlin and H.M. Taylor, *A First Course in Stochastic Processes*, Second Edition, Academic Press, 1975.

[46] R.M. Karp, M.G. Luby and N. Madras. Monte-Carlo Approximation Algorithms for Enumeration Problems. *Journal of Algorithms*, **10**, 1989, 429-448.

[47] J. Kelly, A New Interpretation of Information Rate, *Bell Sys. Tech. Journal*, **35**, 1956, 917-926.

[48] F.P. Kelly, Modeling Communication Networks, Present and Future, Proc. R. Soc. Lond. A (1995) **444**, 1-20.

[49] F.P. Kelly, Notes on Effective Bandwidths, *Stochastic Networks: Theory and Applications*, Vol. 4, Royal Statistical Society Lecture Notes Series, Oxford University Press (1996), 141-168.

[50] J. Kleinberg, Y. Rabani and E. Tardos. Allocating Bandwidth for Bursty Connections, In *Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC)*, 1997, 664-673.

[51] J. Kleinberg, S. Ravi Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. The Web as a graph: measurements, models and methods. In *Proceedings of the 5th Annual International Computing and Combinatorics Conference*, 1999.

[52] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for Emerging Cyber-Communities. In *Proceedings of the 8th WWW Conference*, 1999, 403-416.

[53] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic Models for the Web. In *Proceedings of the 41st Annual Symposium on the Foundations of Computer Science*, 2000.

[54] A. Lempel and J. Ziv, On the Complexity of Finite Sequences, *IEEE Transactions on Information Theory*, **22**, 1976, 75-81.

[55] The Malleable Caches Project at MIT, http://www.csg.lcs.mit.edu/mcache/index.html

[56] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[57] Napster website. *http://www.napster.com*

[58] D. Ornstein, Guessing the Next Output of a Stationary Process, *Israel J. Math.*, **30**, 292-296.

[59] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing order to the Web, *Technical Report*, Computer Science Department, Stanford University, 1998.

[60] C.H. Papadimitriou. *Lecture Notes*, UC Berkeley. Available at `http://www.cs.berkeley.edu/ christos/games/powerlaw.ps`

[61] G. Pandurangan and E. Upfal, Static and Dynamic Evaluation of QoS Properties, In *Proceedings of the 31st ACM Symposium on the Theory of Computing (STOC)*, 1999.

[62] G. Pandurangan and E. Upfal, Static and Dynamic Evaluation of QoS Properties, *Journal of Interconnection Networks*, **1**(2), 2000, 135-150.

[63] G. Pandurangan and E. Upfal, Can Entropy Characterize Performance of Online Algorithms, In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2001.

[64] G. Pandurangan, P. Raghavan, and E. Upfal. Building Low Diameter Peer-to-Peer Networks, In *Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, 2001.

[65] G. Pandurangan, P. Raghavan, and E. Upfal. Using PageRank to Characterize Web Structure, In *Proceedings of the 8th Annual International Computing and Combinatorics Conference (COCOON)*, 2002.

[66] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker. A Scalable Content-Addressable Network in *Proceedings of ACM SIGCOMM*, 2001.

[67] M. Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network, Technical Report, University of Chicago, 2001. *http://www.cs.uchicago.edu/%7Ematei/PAPERS/gnutella-rc.pdf*

[68] J. Rissanen, A Universal Data Compression System, *IEEE Transactions on Information Theory*, **29**(5), 1983, 656-664.

[69] S.M. Ross. *Applied Probability Models with Optimization Applications*, Holden-Day, San Francisco, 1970.

[70] J. Saia, A. Fiat, S. Gribble, A. Karlin and S. Saroiu. Dynamically Fault-Tolerant Content Addressable Networks. in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, March 2002, Cambridge, MA.

[71] S.Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems, in *Proceedings of Multimedia Computing and Networking (MMCN)*, San Jose, 2002.

[72] D.D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules, *Communications of the ACM*, **28**(2), 1985, 202-208.

[73] M.E. Steenstrup (editor). *Routing in Communications Networks*, Prentice Hall, 1995.

[74] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, in *Proceedings of ACM SIGCOMM*, 2001.

[75] Edward Suh and Larry Rudolph, Adaptive Cache Partitioning, *CSG-Memo 432*, Lab. for Computer Science, MIT, June 2000.

[76] M. Taqqu, W. Willinger and R.Sherman. Proof of a Fundamental Result in Self-Similar Traffic Modeling. *Computer Communication Review*, **27**, 1997, 5-23.

[77] L. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, **8**, 1979, 410-421.

[78] J.S. Vitter and P. Krishnan. Optimal Prefetching Via Data Compression, *Journal of the ACM*, **43**(5), 1996, 771-793.

[79] Q. Wang and V.S. Frost. Efficient Estimation of Cell Blocking Probability for ATM Systems, *IEEE/ACM Transactions on Networking*, 1993 **1**(2), 230-235.

[80] WT10g collection draft paper. `http://www.ted.cmis.csiro.au/TRECWeb/wt10ginfo.ps.gz`

[81] J. Ziv and A. Lempel, Compression of Individual Sequences via Variable Rate Coding, *IEEE Transactions on Information Theory*, **24**(5), 1978, 530-536.