STRUCTURAL MINING OF LARGE-SCALE BEHAVIORAL DATA FROM THE INTERNET

Mark Meiss

Submitted to the faculty of the Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in Computer Science

Indiana University

April 2010

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Doctoral Committee

Filippo Menczer, Ph.D.
(Principal Advisor)

Alessandro Vespignani, Ph.D.

Katy Börner, Ph.D.

Minaxi Gupta, Ph.D.

April 30, 2010

Kay Connelly, Ph.D.

Dedicated to my loving wife and partner Heather Roinestad,

without whose support and encouragement

these words could never have been written,

nor the research performed.

Acknowledgements

This work was made possible with the support, encouragement, and active assistance of many generous colleagues at both Indiana University and abroad.

The Advanced Network Management Laboratory, part of the Pervasive Technology Institute at Indiana University, provided not only employment during a long and fruitful period of research, but also many of the computing resources and personal contacts that made this work possible. Special thanks are due to Steven Wallace and Gregory Travis for their support for this research, especially travel to conferences to present portions of this work; David Ripley for his expert technical support and tenacity at problem-solving; Edward Balas for the discussions that first led me to take a graph-theoretic approach to network flow data; and Camilo Viecco for serving as a sounding-board for countless ideas.

My advisor, Filippo Menczer, has taken an active role in supporting this research, including the initial purchase of the HTTP request collection system. His insights and guidance are present throughout, though all errors remain my own. He has shown unbounded patience and kindness throughout the course of this research.

Thanks are also owed to the other members of my committee: Alessandro Vespignani, Katy Börner, Minaxi Gupta, and Kay Connelly. Their suggested readings have helped immensely to

ground this research. I must also thank Dr. Vespignani for invaluable instruction in the statistics of heavy-tailed distributions and the structure and dynamics of complex systems and Dr. Börner for my initial education in the structural analysis of networks. The attractive graphs in this work are a consequence of her course in data visualization; the unattractive ones represent my own lapses in judgment.

I owe further thanks to all those I have collaborated with on portions of this work: Katy Börner, Jean Camp, John Duncan, Alessandro Flammini, Santo Fortunato, Bruno Gonçalves, Filippo Menczer, José Ramasco, and Alessandro Vespignani. Their insights and thoughts have been indispensible.

Special thanks are due to all of the members of the "Networks and Agents Network" within the IU School of Informatics, whose feedback and suggestions have helped to hone my presentation skills and inspired many avenues of research. Dr. Menczer's support for this group during even the busiest semesters has been a vital part of this research.

Dr. Camp, the network engineers of Indiana University, and the Internet2 staff have my thanks for their assistance in arranging access to the data sources used in this work. Damon Beals, Dave Hershberger, and John Stigall of UITS Network Operations are owed extra thanks for their assistance in setting up collection systems in inconvenient places at inconvenient times.

Besides the Advanced Network Management Laboratory and the School of Informatics, material support for portions of this research was provided by NSF awards 0348940, 0513650, 0705676; the Institute for Scientific Interchange in Torino, Italy; the Institute for Information Infrastructure Protection research program, funded by Award 2003-TK-TK-0003 from the U.S. DHS, Science and Technology Directorate; Google, Inc.; National Institute of Health grant NIH-1R21DA024259; and Project 233847-Dynanets of the European Union Commission.

This work would have been impossible without the continual support of my family and friends. My parents and my brother deserve my thanks for not only their love and support, but raising me to see scientific inquiry as a way of life. My closest friends, Heather and Dennis Pund, have always encouraged me to maintain my curiosity and express my thoughts in honest, direct terms. Finally, my wife, Heather Roinestad, besides offering many insights through her own technical expertise, has provided more patience and love than I can express; it is to her this work is dedicated.

Abstract

As the Internet becomes ever more pervasive in the lives of hundreds of millions of people, our understanding of its physical structure has outpaced our understanding of the dynamic patterns of traffic generated by its users. This work aims to develop a better understanding of the structure of Internet traffic in a manner consistent with individual privacy and computational constraints. I first examine network flow data from the Internet2 network, using it to form "behavioral networks" based on the flows attributable to specific network applications. The heavy-tailed distributions associated with these networks suggest unbounded variance and poorly defined means in distributions of user behavior. However, a novel application of hierarchical clustering to similarity data derived from these networks makes it possible to classify network applications robustly based on their observed behavior. I then focus on Web traffic, using a large collection of HTTP request data to build a weighted subset of the Web graph. Analysis of this weighted graph reveals more heavy-tailed distributions and the presence of a large body of stationary traffic. The traffic data are also shown to contradict key assumptions of the random surfer model used by PageRank. I conclude with the development of ABC, an behaviorally plausible agent-based model of Web traffic that incorporates backtracking, bookmarks, and a sense of topical locality. The ABC model is shown to approximate real user activity more accurately than PageRank on both artificial and empirically generated graphs.

Contents

List of Tables

List of Figures

1

Introduction

1.1 Evolution of the Internet

In 1969, few people took notice of the beginning of the worldwide network of networks we now know as the Internet [97]. This was hardly surprising in the year that humans first walked on the surface of the moon; the infant network, ARPANET, connected only four computers, and relatively few people had any direct experience with computers. A tiny company called Intel was just beginning its development of the first microprocessor [21], and no one had a computer in their home or their car, let alone their purse.

People took a number of facts about consumer technology for granted. If you missed your favorite show on television, that was just too bad; you had to hope for a rerun. If you needed to communicate a written message more quickly than mail could offer, you sent a telegram. If you needed to speak with someone far away, you placed a telephone call and spoke to their disembodied voice. If you wanted to play a game with somebody far away, you signed up for a service such as "chess by mail."

Forty years later, each of these assumptions has been demolished by the innovations that attracted so little attention at their start. Who worries about missing a television show when it can be captured by a digital video recorder, traded on file-sharing networks, or watched as a streaming download from Hulu or YouTube? Telegrams seem laughable to a generation for whom e-mail is too slow and inconvenient compared to a series of rapid-fire instant messages. The telephone has yet to vanish (though it has unplugged itself from the end table in the parlor and found its way into our pockets) but many have turned to Internet-enabled video conferences as a way of saving money over a traditional long-distance phone call. Finally, many now scoff at the notion of playing a game either by mail or against a single opponent, preferring to spend their leisure hours playing in real time with hundreds of thousands of other people in a shared world such as World of Warcraft.

Even as the growth of the Internet has changed the life of the average person, it has transformed the world of academic research to a far greater extent. Network applications such as e-mail and shared document editors have made it possible for scientists half a world apart to collaborate actively and share their results in real time. Researchers are able to share their data sets with others without negotiating traveling expenses or the cost of duplication. Their publications are more widely available than anyone could have imagined in 1969: a hand-written reprint request followed by a wait of several weeks has been replaced by a single web click. The discovery of prior results is now aided by search engines and hyperlinked networks of citations as much as tables of contents and word of mouth.

Something curious has happened in the midst of all this change. The global network that enabled these transformative changes in both everyday life and the world of academia has, because of the very magnitude of its influence, become an object of study in its own right. Indeed, with well over a billion people online [74], and with the world's financial institutions, governments,

and militaries depending on its continued operation, we can ill afford *not* to study this network of networks. How can a complex system with no single controlling authority, no explicitly defined goals, and no centralized management, grow more quickly and become a more pervasive influence than networks that were designed with clear intentions and goals? Because the Internet has grown exponentially and without central direction, we cannot know its size, structure, and dynamics of growth *a priori*; we must study these properties experimentally.

The study of any complex system involving many millions of people around the world is inherently of interest to social scientists. Not only is the physical network the product of shared efforts from thousands of engineers from many cultures, but the traffic that moves across it represents behavioral interactions of more than a billion people around the world. A study of behavioral data that might once have involved a hundred individuals can now examine hundreds of thousands or even millions of Internet users. By studying what people do online, we learn more about ourselves and our societies.

There is also a more practical motivation for understanding the structure, growth, and dynamics of the Internet. Only by measuring its properties and relating these measurements to theoretical results in complex systems are we able to make meaningful predictions about the future growth of the network—and, more importantly, its vulnerabilities. Does the topology of the Internet create weak spots that determined individuals can exploit to disrupt its operation? Does the interaction of users on the network follow patterns that allow us to combat the spread of malicious software with targeted immunization of vulnerable systems? Can we relate the physical structure of the network to the structure of its traffic in ways that increase its efficiency and make it more robust?

1.2 Evolution of the Web

Any meaningful study of Internet traffic requires us to acknowledge that the Internet of 2009 exhibits a feature not present at its inception: a dominant application, in the form of the World Wide Web. Though the Internet had already been functioning for twenty years before the beginning of the Web in 1991, the growth of the Web has been so explosive that we must also understand its structure and dynamics to understand those of its host network. In fewer than twenty years, the Web has grown from a single page at CERN to the point that Yahoo! discovers over one trillion URLs through its web crawlers [45]. The number of Web servers has similarly skyrocketed, and technology such as virtual hosting makes it possible for the number of Web sites to exceed the number of computers physically attached to the network. Estimates vary as to the proportion of network traffic devoted to the Web—and, as we shall see, exactly what constitutes "Web traffic"—but for many millions of users, the Web and the Internet are functionally the same. Understanding what they do online is the same as understanding how they navigate the Web.

In considering the growth and structure of the Web, we cannot forget that the content of the Web has been evolving just as rapidly as its structure. Little has remained static about the Web's content from its inception through today other than the notion of hyperlinking and the use of documents containing a large number of angle brackets. The earliest Web pages hosted primarily static information about people and places, but it was not long before the Web began evolving from a broadcast medium in which information is only viewed, into an interactive medium in which it is shaped by many hands. Message boards, discussion forums, comment forms, interactive games, blogs, and wikis have all shaped our shared understanding about what the Web is even as the underlying protocols have remained largely the same. The difference between the first home page at CERN and Google Docs is slight from the perspective of HTTP, but enormous when we seek to understand the structure and function of the Web *as it is used*.

A dominant feature in the growth of the Web has been steady evolution in the ways in which users locate the information it provides. It became clear early on that simply clicking on links would not offer sufficient means of navigating a potentially unbounded information space, leading first to the development of web directories and then to search engines. At first, these search engines directed users to pages based entirely on their textual content, but exponential growth in content and the commercialization of the Web made this untenable: the rank of a page in search results became not a measure of the quality of its content, but its quantity of search terms (possibly in no meaningful arrangement). Google's introduction of link-based analysis for ranking pages transformed the Web in two critical ways. Most importantly from the perspective of users, it helped to "rescue" the Web from commercial spam and made it once again possible to locate useful information in a sea of billions of pages. Even more importantly from the perspective of Internet researchers, it created a novel relationship between the *structure* of the web and its content. The link structure of the Web and the behavior of its users have become so interrelated that neither can be studied in isolation. Indeed, we are now witnessing significant debate on the extent to which, by *examining* the link structure of the Web, Google and other search engines actually *determine* its structure [111].

1.3 Themes

These trends in the evolution of the Internet and the Web all help to motivate a central theme of the work presented here: the physical structure of a network cannot be understood separately from the behavior it supports. While there is much to be learned from studying the physical arrangement of network connections or the pattern of hyperlinks connecting Web pages, these structures evolve in direct response to user behavior: the network is shaped by the actions it makes possible. We achieve greater success in characterizing user behavior, identifying malicious activity, and planning

future infrastructure, by understanding network applications not as the software expression of a particular protocol, but rather based on the patterns of traffic generated by their users. For example, the NNTP protocol underlying USENET has not changed in the past twenty years, but the USENET of 2009 and the USENET of 1989 are entirely different applications in terms of their audience, their structure, and their impact on the network. In a fast-changing digital world in which almost any protocol can be tunneled through any other with clever enough means, we can best understand what an application *is* by examining what it *does*.

Equally important in this work is another theme, this one derived from the example of Google and other link analysis systems: the patterns of interaction among users on the Internet are as important for user modeling, network security, and network planning as the actual payloads themselves. The context of a node in a network may offer just as many clues to its identity and function as examining the content of its conversations. As we shall see later, in some cases, we can actually learn more about the role of a host on the Internet by studying its relationship with other hosts than by inspecting its traffic. Furthermore, study of the aggregate pattern of interaction among hosts helps us to understand the significance of the observed behavior of a single host. We can move from saying that a host is behaving anomalously to saying that it is doing so *within its social context*—and even show that nature of some online communities makes the notion of an "anomaly" difficult to define.

This work also emphasizes the practicality of taking a pattern-based approach to analyzing online behavior rather than attempting to characterize malicious activity through deep packet inspection. High-speed network interfaces operate at data rates that far exceed the pattern-matching ability of general-purpose processors, especially because reasonably sophisticated attack detection requires matching regular expressions, processing fragmented packets, and reassembling TCP streams. A robust, stateful packet inspection system must have more processing power than the

routers that forward packets—but these very routers are the first devices to incorporate breakthroughs in packet processing technology. Believing that packet inspection systems will eventually catch up to the data rates of modern wide-area networks requires believing that we will come to value network security as a more precious good than the network itself, which is economically unlikely.

The overhead associated with gathering information about the patterns of connections among users, computers, and Web sites, is much smaller. Network flow records, the basis of much of the analysis presented in this work, can describe network transactions ranging from a simple echo request to a multi-megabyte file transfer with the same fixed-length 48-byte record. This is not to say that packet analysis has no place in a scalable network security system: a pattern of suspicious activity cannot constitute proof of wrongdoing in the same way as actual copies of the offending traffic. However, information gleaned from the network of interactions can certainly suggest which Internet hosts are most deserving of closer inspection, and, as I will argue later, can do so based on information not present in the packets themselves. A list of suspicious hosts derived from structural mining of traffic patterns can be prioritized and trimmed until it is within the computational reach of sophisticated packet analysis systems, making behavioral analysis as described in this work a scalable and practical front-end for a hybrid network security system.

A final theme which will emerge periodically throughout this work is that structural analysis of behavioral data can be performed in a way more respectful of user privacy than is standard practice for other sources of network traffic data. Considering the properties of someone's position in the global network of e-mail users does not require reading their mail. Learning that two applications are correlated in ways that imply similarity of use even though their protocols differ significantly does not depend on intercepting users' actual traffic. Exploring the network of Web sites that are visited in combination with each other can be done without instrumenting users' browsers or

retaining their personal identities. The techniques of structural data mining can be applied in ways that abuse privacy—the ill-fated TIPS database is testimony to this [55]—but they may offer new means of protecting privacy by offering investigators the means to establish probable cause for further investigation without intruding on legitimate activity.

1.4 Organization

Before these themes can be explored in a technical and concrete way, it is necessary to provide appropriate background information. In Chapter 2, I discuss the basic principles of graph theory as they relate to analysis of network data. With this foundation, I describe the salient features of modern data networks and the varieties of usage data they make available. The first such variety consists of network flow records, a form of *indirect sampling* of activity that provides a scalable means of investigating network traffic in general. I then narrow my focus to the dominant application on the Internet—the World Wide Web—and discuss the *direct sampling* of data sources particular to the Web, which will serve as the basis of the analysis described in Chapter 4. I briefly outline the graph structures that can be derived from these data sources and introduce the structural data mining techniques that are tractable for such large data sets. Finally, I offer a survey of related work in the field and situate the present work in the context of these efforts.

Chapter 3 discusses the gathering and analysis of general Internet traffic as represented by network flow data. I begin by discussing the properties and dimensions of the particular data sources used and the means used to collect the data. I also discuss anonymization requirements placed on the data, as well as the processes used to derive various graph structures from the raw flow data. Collection and analysis of such large data sets required the development of a number of custom tools for working with graph structures too large to be held in main memory, which I outline

briefly.

Following this discussion of data gathering, I present an analysis of the structural properties of the graphs derived from network flow data. I describe the extremely broad distributions of degree, strength, and weight found to be typical of Internet traffic in general and some classes of network application in particular. I also describe some of the scaling relations among these distributions and their significance. Finally, I comment on some difficulties of applying more complex analysis to these behavioral network, especially clustering measures and spectral analysis.

Besides the static structural properties of network flow data, I also consider its temporal aspects. In particular, I present a longitudinal analysis of flow data across several years, showing that the structural properties presented earlier are remarkably consistent even as the data source has evolved in both volume and commercial character. I also explore the extent to which these graph structures can be used as predictors of future network activity.

The chapter continues with a discussion of the applications of structural data mining on network flow data. I describe in detail how a similarity graph based on the joint use of applications can be used to construct a behavioral taxonomy of network applications and then identify the function of unknown applications without access to the payloads of individual packets. I discuss how the broad distributions found in traffic graphs limit the performance of threshold-based anomaly detection, making them unsuitable as metrics for identifying anomalous activity. Finally, I comment on the implications of my findings for models of network traffic, application design, and network capacity planning.

The discussion of network flow data would be incomplete without considering potential sources of bias, particularly because the analysis described is based on sampled data. The chapter thus concludes with an discussion of possible sources of bias in my data sets, focusing on the effects of packet sampling, limited collection capacity, malformed data, and implicit trust of unreliable

sources. I then present Flow Generation Language (FGL), a system I designed to aid the process of constructing known patterns of network traffic, injecting them into the network, and examining their reflection in network flow data. I describe a study involving the use of FGL to introduce traffic data with long-tailed distributions into the Internet2 network and present its findings. This study yields two results: first, that packet sampling on the current generation of backbone routers indeed appears to obey a uniform random distribution; and second, that the properties described in this dissertation are robust with respect to packet sampling given sufficient levels of aggregation. I offer theoretical support for these findings and suggest ways in which some of the other potential biases present in the data might be investigated.

Chapter 4 moves from the analysis of network traffic in general to a more focused analysis of Web traffic ("click") data. After a brief discussion of the primary sources of Web usage data, I offer a concrete description of my own data source, which consists of HTTP requests harvested directly from backbone network interfaces. I describe the details of collecting this data and the advantages and disadvantages of this particular source of information. Because this data source was also too large for analysis with existing general-purpose tools, I briefly describe the specialized tools used to facilitate my study.

After establishing the nature of the data available, I present my findings regarding the structural properties of the weighted subset of the Web graph described by the click data. I show how the observed degree distributions agree with previous research on the link structure of the Web, but the strength and weight distributions demonstrate that the random surfer model of PageRank is a poor predictor of actual Web traffic. I identify instances in which the random surfer model assumes a uniform probability that turns out to be better described by a broad, scale-free distribution. I also present results on the temporal properties of the click data, showing that many of the distributions of the traffic graph remain quite stable from day to day, but that some properties obey daily and

weekly cycles. I describe how a surprisingly large proportion of Web traffic turns out to be static, with even stronger daily and weekly cycles present in the structure of the graph.

I then describe the analysis of a secondary data set in which the click streams of different users were kept distinct from one another. Besides replicating the basic properties of the larger body of click data, this set provided access to a number of per-user distributions, which I describe together with their implications for existing Web research. I also describe the results of attempting to segment individual users' click streams into sessions, leading to the conclusion that timeout-based notions of Web sessions should be discarded in favor of session trees built using referrer data from the original requests. I present the basic properties of these session trees and demonstrate how they overcome the shortcomings of previous definitions.

Finally, I describe some applications of structural mining of Web click data. Key among these is the construction of improved models of Web surfing better able to predict Web traffic than the random surfer model. I describe a series of models that achieve increasingly better performance at capturing the properties of empirical Web traffic than the random surfer model. The first model simply demonstrates that the heterogeneity of traffic handled by the outgoing links of a Web page is insufficient to account for the failure of PageRank to reproduce empirical traffic distributions and that some form of state is necessary for more robust modeling. I then introduce BookRank, a stateful model developed in conjunction with my collaborators that retains lists of previously visited pages (i.e., bookmarks) and is able to approximate real-world distributions of site popularity and time between visits more accurately than the random surfer model. Finally, I describe ABC, an improved version of BookRank that incorporates an energy-based model based on theories of topical locality in the Web and which is capable of generating logical Web sessions that reproduce key features of empirical browsing data.

Finally, in Chapter 5, I offer a summary of the key results of this research, with particular attention to the findings which are most novel and useful to other researchers in the field. I examine the implications of my findings and suggest possible avenues of future research for further exploration. I describe the future work I intend to pursue in structural data mining, which will involve more work in heuristic estimation of network properties, agent-based models, and traffic forecasting.

2

Background

The work described in this dissertation combines many aspects of both graph theory and data networks. Although these fields naturally complement one another, discussions involving their combination are often made confusing by overlapping sets of terms that are also common in casual conversation. Sometimes, "a node in the network" refers to a general-purpose computer with a connection to the Internet; at other times, it may refer to a vertex in an abstract graph structure. A further complication is that many disciplines have reinvented topics in graph analysis, leading to there being several names in active use for a variety of concepts. In such an environment, it is unwise to assume that terms, even fairly basic ones, from graph theory will be concrete and familiar to experts in data networks, and vice versa. A primary aim of this chapter is thus to introduce the requisite features of each field in a manner accessible to practitioners of the other, with apologies to any reader already experienced with both. This is done in Sections 2.1 and 2.2.

After introducing this foundational material, I discuss in Section 2.3 the principal ways in which data networks can be instrumented to provide behavioral data for both the Internet as a whole and the Web in particular. Each potential source of data comes with attendant advantages and disadvantages that must be evaluated with respect to analytical needs and the difficulty of collecting

them. I explain my motivations for selecting the primary sources I have: network flow data in the case of network traffic as a whole, and HTTP request data in the case of the Web.

In neither the case of network flows nor that of Web requests does incoming data already describe a graph structure. Their direct representation is that of events or transactions, not complex graph structures. In Section 2.4, I describe how these time series data sets can be used to derive a variety of graph structures, and I identify those used in the present work.

A wide variety of analytical techniques can be applied to graphs representing behavioral data, and their practicality is influenced by both the structure and size of the graphs under consideration. In the case of the present analysis, the graphs often contain on the order of ten million nodes, making only a limited set of techniques applicable to the empirical data. In Section 2.5, I identify techniques that are appropriate to the analysis of extremely large graphs and that form the basis of this work.

My work is hardly the first to examine graph structures formed from behavioral data, nor is it the first to search for deeper meaning in patterns of network traffic. Its unique contributions lie in the application of structural data mining techniques to the extremely large behavioral networks we are able to form from Internet data, and in the insights into behavior and modeling that result from that application. In Section 2.6, I discuss a variety of related efforts in the past and situate my own work with respect to these projects.

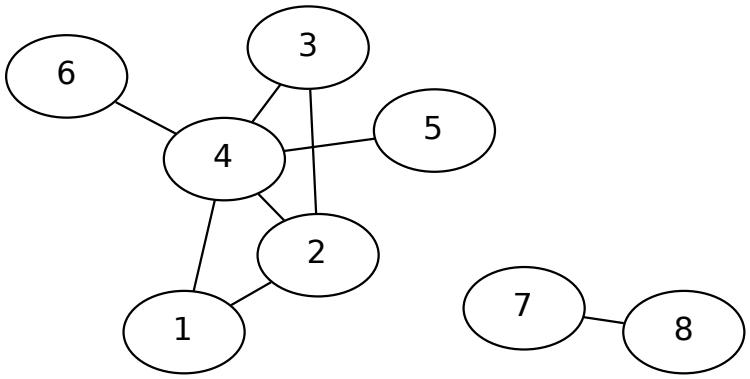


Figure 2.1: Example of an undirected graph containing only nodes and edges.

2.1 Graph theory

Undirected graphs

However familiar the concept may be, it is worthwhile to begin with the most basic definition of a graph in order to establish consistent nomenclature and avoid later confusion. We define an undirected graph G as a pair of sets (N, E), where N is a set of *nodes* or *vertices* and E is a set of *edges*. The set of edges is a symmetric binary relation over the nodes; that is, $E \subset N \times N$, and $\forall(a, b) \in E, (b, a) \in E$. An example of an undirected graph can be seen in Figure 2.1. In this work, I use the terms node and vertex interchangeably, with the caveat that *node* is sometimes used to refer to a generic physical device in a data network, such as a router or personal computer.

Note that the definition above excludes *multigraphs*, in which multiple edges may share the same endpoints. Although multigraphs are a natural representation for time series data in which events may repeat, they present additional challenges for both storage and processing. In this work,

I prefer to handle duplicate edges through aggregation and the use of weighted graphs.

If there exists a sequence of n edges $(a_1, b_1), \ldots, (a_n, b_n)$ from E such that $b_i = a_{i+1}$, we call that sequence a *path* of length n and say that nodes a_1 and b_n are *connected*. If there is a path of length 1 between a and b, we say they are *directly connected*.

The notion of connectedness gives us an intuitive method of partitioning the nodes in a graph, in which two nodes a and b are in the same equivalence class if and only if there exists a path between a and b. We refer to such equivalence classes as *connected components* of the graph. In the case of random graphs and many graphs arising from natural processes, including those considered in this work, once the edge density $\frac{|E|}{|N|^2}$ becomes large enough, there arises a single connected component that contains a majority of the nodes in the graph [120].

One intuitive means of representing the information in E is an *adjacency matrix*. The adjacency matrix A of a graph is an $|N| \times |N|$ matrix in which $A_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise. In the case of an undirected graph, this matrix is symmetric, since E is a symmetric relation over the nodes. The adjacency matrix provides a convenient abstraction for formally defining graph properties and features prominently in some graph algorithms and analytical techniques discussed in this work, such as PageRank and spectral analysis.

We often speak of the "size" of a graph without making it clear whether we refer to the cardinality of N or of E. In theory, this can make a considerable difference: $|N|$ is the dimension of the adjacency matrix and $|E|$ is the number of non-zero entries it contains, so $|E|$ is potentially $O(|N|^2)$. In the present work, these "dense" graphs seldom arise, and the number of edges is roughly proportional to the number of nodes, allowing us to use either quantity to inform discussion of the size of a graph. Another aspect of graph size that causes occasional confusion is the notion of a "large" graph: every academic discipline that includes the study of complex networks has its own notion of what constitutes a large data set, with no general agreement across fields. In this work, I avoid

saying that a graph is large unless it contains enough nodes and edges so as to make direct visualization impossible; its storage requirements are comparable to or in excess of the memory available on contemporary computers; and algorithms of complexity $O(n^2)$ or higher are computationally intractable on a single workstation. At the time of writing, this threshold is on the order of 100,000 nodes.

Another phenomenon displayed by many graphs discussed in this work occurs when every node in the graph has at least one incident edge—that is, $\forall a \in N, \exists b \in N \mid (a, b) \in E$. Under these circumstances, we can omit N from our representation of the graph, since it is defined implicitly by the edges in E. This has practical benefits for a number of processing techniques, making it possible to process a large graph in serial fashion as a list of edges.

Directed graphs

Although the graphs discussed so far are able to model many real-world data sets, they discard important information from many others. If a user downloads a Web page, the relationship between their computer and that of the Web server is not symmetric: clearly, information is passed *from* the server *to* the user's computer. Similarly, when Web page A links to page B, it is not necessarily the case that B links back to A. We can resolve this difficulty by removing the restriction that the edge relation be symmetric, giving us a *directed graph* or *digraph*. As shown in Figure 2.1, the edges in such a graph are often drawn using arrows to indicate that the edge (a, b) goes *from* node a *to* node b. We refer to node a as the *initial*, *source*, *tail*, or *origin* node of the edge; and node b as the *final*, *destination*, *head*, or *target* node.

Because the edge relation of a directed graph need not be symmetric, the same is true of its adjacency matrix. This complicates the definitions of a number of graph properties, especially the notion of connectivity explored in the previous section: the existence of a path between nodes a

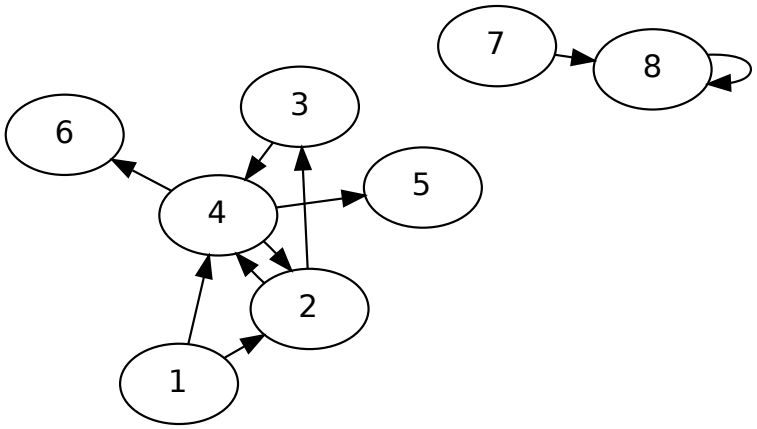


Figure 2.2: Example of an directed graph containing only nodes and edges; the arrows indicate the direction of the edges.

and b does not guarantee the existence of a reciprocal path from b to a. For any node a in the graph, we can refer to the set of nodes R for which there exists a path beginning at a and terminating at a node $b \in R$ as the *reach* of a. This is simply an expression of the intuition that if there exists a path from a to b, one can arrive at b from a.

A graph that would tend to have a single connected component if it were undirected may instead exhibit "bow tie" topology, as illustrated in Figure 2.1. In this arrangement, the central region is a *strongly connected component* (SCC) which corresponds to components in an undirected graph: for any two nodes a and b selected from the strongly connected component, there exists at least one path from a to b and one from b to a, though the two may be dissimilar. The left end of the bow tie contains nodes *from* which one can reach the SCC, but not the converse; these are "source" nodes for the graph. Similarly, the right end of the bow tie contains nodes that can be reached from the SCC but cannot reach it themselves; these are "sink" nodes for the graph. There are also "tube"

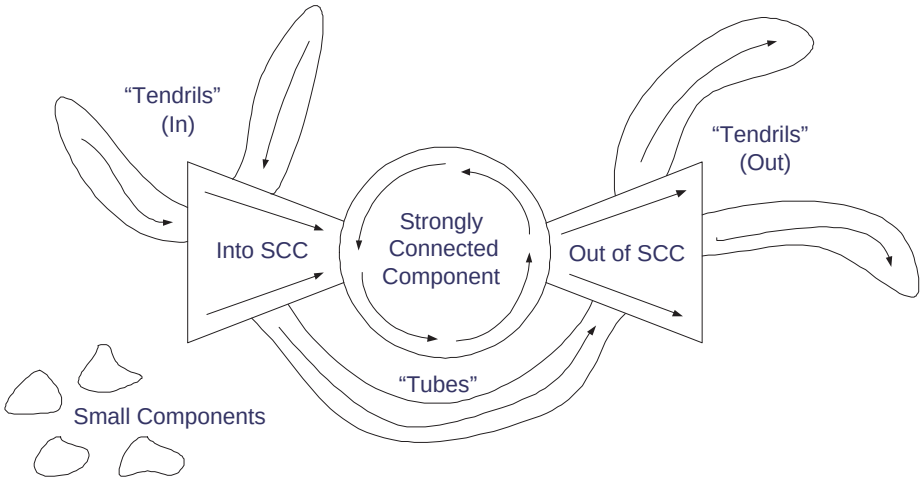


Figure 2.3: Illustration of "bow tie" topology in a large directed graph. This topology serves as a general framework for understanding the relative number of nodes and edges in each role, and has been applied often to discussions of the Web [30].

nodes that make it possible to travel from the source to the sink nodes without passing through the SCC and nodes unconnected to the bow tie at all, but these may represent a small proportion of the total. The most well-known example of such a graph topology is the link structure of the Web, first described in [30].

Weighted graphs

Another feature that can be added to either directed or undirected graphs is that of *weights* for the edges. In a weighted graph, the relation E becomes a function $N \times N \mapsto \Re$, where $E(a, b)$ is the *weight* of the edge from node a to node b. If there exists no edge from a to b, we often say that $E(a, b) = 0$.

Weighted graphs arise naturally in a variety of situations. In the case of an undirected graph, weights may reflect a measure of similarity between the objects represented by the nodes; this similarity is not a consequence of the structure of the graph, but represents an additional source of

information about the nodes and their relationship to one another. In the case of directed graphs, weights are often a rudimentary form of dynamic information, such as a rate of activity or volume of flow. For example, we can add weights to the edges in the link graph of the Web to reflect the number of times users actually traverse each link.

As mentioned earlier, weighted graphs are especially useful for avoiding the complications associated with multigraphs. Instead of allowing there to be three edges from node a to node b, we can instead aggregate them into a single edge with a weight of three. In the case of graphs whose edges are derived from events in time-series data, this can result in the loss of information about the exact timing of activity, but for many purposes aggregate volumes or rates of activity are sufficient. The present work uses this form of aggregation extensively; however, in Chapter 4, I do present some methods of comparing graphs that represent aggregated slices of time-series data.

We can extend the notion of the adjacency matrix to a weighted graph simply by making its entries correspond to the weights of the edges; that is, let $A_{ij} = E(i, j)$. This alternate formulation is sometimes referred to as the *weighted adjacency matrix*; however, since most of the graphs discussed in this work are weighted, I drop the qualifier and use the term *adjacency matrix* to refer to the weighted version. When I must refer to the version that contains only zeros and ones as entries, I call this the *binary* or *boolean* adjacency matrix.

Bipartite graphs

The graphs discussed thus far have all shared the property that the nodes are of a single type. However, the graphs arising from many real-world data sources have the property that the set N can be partitioned into two disjoint sets N_A and N_B such that $(a, b) \in E$ implies $a \in N_A$ and $b \in N_B$. In other words, we have two distinct classes of nodes, and every edge connects a member of one class to a member of another class. In this case, we say that we have a *bipartite graph*, which we can

represent as a triple $(N_A, N_B, E \subset N_A \times N_B)$. We can extend this notion to more than two classes of nodes, but these *multipartite* graphs are outside the scope of this work. Note that whether a graph is unipartite or bipartite is independent of whether it is directional or weighted; any of these features can exist in isolation.

The adjacency matrix of a bipartite graph introduces some complications: whereas the dimensionality of the adjacency matrix is simply $|N| \times |N|$, for a bipartite graph, it is $|N_A| \times |N_B|$, with $|N_A| \neq |N_B|$ in general. In other words, the adjacency matrix is no longer square, making it impossible to invert, take powers of, and so forth. When this situation arises, we can consider a bipartite graph as simply being a unipartite graph with $N = N_A \cup N_B$ and a more sparse edge relation. The adjacency matrix then has dimensionality $|N_A + N_B| \times |N_A + N_B|$ and is again square.

Bipartite graphs can also be analyzed by deriving unipartite "co-citation" graphs from each class of node and examining them instead. If we have a bipartite graph $G = (N_A, N_B, E)$, then let $G_A = (N_A, E_A)$ be a unipartite graph with the property that $(a_i, a_j) \in E_A$ implies that $(a_i, b) \in E$ and $(a_j, b) \in E$ for some node $b \in N_B$. Similarly, let $G_B = (N_B, E_B)$ be a unipartite graph with the property that $(b_i, b_j) \in E_B$ implies that $(a, b_i) \in E$ and $(a, b_j) \in E$ for some node $a \in N_A$. That is, nodes in G_A are adjacent if they both share edges with some node in N_B, and nodes in G_B are adjacent if they share edges with some node in N_A. These derived graphs have square adjacency matrices and can be analyzed in the same manner as any other unipartite graph. Unfortunately, these unipartite projections are often much denser than the bipartite graphs from which they originate, especially in the presence of heavy-tailed degree distributions. In the case of the data sets involved in this work, the co-citation graphs are too far large to store and analyze using existing tools.

In some cases, a graph may be "almost" bipartite in the sense that a small number of nodes in the graph share edges with nodes of either class. In other words, the sets N_A and N_B are not

disjoint, but the quantity $\frac{|N_A \cap N_B|}{|N_A \cup N_B|}$ is close to zero. This situation arises when the original source of data identifies the classes of the nodes and indicates that some nodes are members of both classes. For example, a graph of Web activity may have two classes of nodes: Web clients and Web servers. However, nothing prevents a single computer from running both Web browser and Web server software, so a relatively small proportion of nodes in the graph are both clients and servers. These nearly bipartite graphs do not require special analytical techniques, but the degree and nature of the overlap between the classes of nodes can be of interest. This theme will be explored more fully in Chapter 3.

Degree and strength

We now consider some of the basic measures that can be associated with the nodes of a graph and are tractable to compute even for very large graphs.

The *degree* of a node in an undirected graph is simply a count of the number of nodes with which it shares an edge. Put formally, if $G = (N, E)$ is an undirected graph, then the degree $k(a)$ of a node $a \in N$ is equal to the cardinality of the set $\{b \,|\, (a, b) \in E\}$.[1] This definition can be simplified by stating it in terms of the boolean adjacency matrix: $k(a) = \sum_{j=1}^{|N|} A_{aj} = \sum_{i=1}^{|N|} A_{ia}$. Note that $\sum_{i=1}^{|N|} k(i) = 2|E|$; that is, the sum of the degrees of the nodes in the graph is equal to twice the number of edges.

We can extend the notion of degree to directed graphs by splitting it into two related quantities: *in-degree*, which is a count of the number of edges incoming to a node; and *out-degree*, which is a count of the number of edges outgoing from a node. In terms of the adjacency matrix, we define the

[1] Although the letter k does not appear anywhere in the word "degree," it is the customary designation in equations.

in-degree of node a as $k_{in}(a) = \sum_{i=1}^{|N|} A_{ia}$ and the out-degree as $k_{out}(a) = \sum_{j=1}^{|N|} A_{aj}$. The in- and out-degrees of any particular node in a directed graph are not necessarily related, but it is worth noting that $\sum_{i=1}^{|N|} k_{in}(i) = \sum_{i=1}^{|N|} k_{out}(i) = |E|$, implying that the mean values of in- and out-degree are identical even if the distributions are very dissimilar. In this work, I adopt the frequently-used notation $\langle x \rangle$ to denote the mean value of the distribution x. Thus, we can simply write $\langle k_{in} \rangle$ = $\langle k_{out} \rangle$.

The degree of any particular node in a graph is generally of less interest than the distribution of degree values for all the nodes in the graph. The degree distribution can take many different forms and is strongly dependent on the mechanisms that generated the graph, a point that will arise later in characterization of both the network flow and Web click data sets.

The *strength* property is a straightforward extension of the notion of degree to weighted graphs. The strength of node a is simply equal to the sum of the weights of its edges. In an undirected graph, we can formally define this in terms of the (weighted) adjacency matrix: $s(a) = \sum_{j=1}^{|N|} A_{aj} = \sum_{i=1}^{|N|} A_{ia}$. For directed graphs, we follow the example of degree and divide strength into two quantities: *in-strength*, defined as the sum of the weights of the incoming edges; and *out-strength*, defined as the sum of the weights of the outgoing edges. Formally, we write $s_{in}(a) = \sum_{i=1}^{|N|} A_{ia}$ and $s_{out}(a) = \sum_{j=1}^{|N|} A_{aj}$.

Because the weights of a graph are usually unconstrained, the sum of the strengths of the nodes in the graph lacks any inherent significance beyond that provided by the original data source. For example, if weights represent the number of bytes of data transferred, the sum of strengths of all nodes would be a measure of total data transfer. It is still the case, however, that the mean values of in-strength and out-strength are identical: $\langle s_{in} \rangle = \langle s_{out} \rangle$.

The form of the distribution of the strengths of the nodes is again of great interest. In the case

of strength, the distribution is once again influenced both by the generative process for the graph, but it is also affected by the underlying distribution of weights.

Assortativity and scaling relations

While the degree and strength distributions of the nodes of a graph are of interest in themselves, their relationships to each other can also be of significance. A key example is the notion of *assortativity* with respect to degree, which considers whether nodes of high degree are more likely to connect to nodes of low degree or other nodes of high degree. If high-degree nodes tend to link to other high-degree nodes (high assortativity), the graph may have a densely connected core of high-degree nodes with a haze of low-degree nodes around the periphery: a "tennis ball" shape. If high-degree nodes tend to connect to low-degree nodes (low assortativity), the graph may have a more modular shape, consisting of a network of sparsely connected hubs: a "clump of burrs" shape.

The complex networks community has yet to reach consensus for single formal measure of assortativity with respect to degree. One common approach is to examine the mean degree of the neighbors of a node as a function of the node's degree, as proposed in [126]. In this work, I take a closely related approach: for each edge in the graph, I consider the degree of the originating (source) node and that of the target (destination) node, then plot the mean destination degree as a function of source degree. An analogous procedure can easily be applied to strength instead of degree.

The exact form of this function is of secondary interest compared to whether the function is increasing or decreasing. Previous research has found that graphs deriving from social interactions tend to have increasing functions (high assortativity), whereas graphs derived from biological data

and technological designs such as the Internet tend to have decreasing functions (disassortativity) [120]. This makes it particularly interesting to examine assortativity in the context of graph structures derived from behavioral data that takes place on the disassortative substrate of the Internet. Which phenomenon will dominate: the assortativity of social interaction, or the disassortativity of design?

Directed graphs add complexity to this discussion, since we can treat in-degree and out-degree as separate quantities. This can be done in two ways. First, we can examine whether nodes with high out-degree are more or less likely to connect to nodes with high in-degree. This is closely related to the discussion above and is particularly interesting with respect to the structure of the Web. In the Web graph, we commonly refer to pages with a high out-degree as *hubs* and those with a high in-degree as *authorities*; the question of assortativity becomes one of asking whether hubs are more or less likely to link to authorities. Second, we can examine whether nodes that have high in-degree are also likely to have high out-degree. If we again cast this in terms of the Web, the question becomes whether Web pages are generally either hubs or authorities exclusively, or whether a substantial number of pages are both.

Notions of assortativity carry forth from degree to strength in a straightforward way, but the underlying analogies may change. When we consider assortativity of degree in the Web graph, we are examining the link structure of the Web. When we consider assortativity of strength in a subset of the Web graph weighted by traffic, we are examining actual behavior on those links. The question becomes not whether hubs link to authorities, but whether the links between hubs and authorities are actually used by real Web surfers. I will revisit this association of degree with underlying structure and strength with observed behavior on several occasions.

The relationship between strength and degree is also of interest. We will observe extremely heavy-tailed distributions for both degree and strength in a number of the graphs discussed in this

work, but examination of either distribution in isolation does not inform us of any clearly defined scaling relation that may exist between the two. Strength will naturally tend to be a increasing function of degree. Of greater interest is whether strength scales *sublinearly* with respect to degree, implying a node's limited resources are divided among an ever-growing number of neighbors; or whether it scales *superlinearly*, implying some sort of synergy among neighbors, so that their whole becomes more than the sum of their parts.

Clustering

A final property of interest that can be calculated for moderately large graphs is *clustering*, which is most often understood as a measure of transitivity in a graph: if node a shares an edge with node b and node b shares an edge with node c, then how likely is it for nodes a and c to share an edge directly? We can measure this by assigning a *clustering coefficient* C to the entirety of the graph, which we define as the ratio of the number of triangles in the graph to the number of instances in which a node has two distinct neighbors, multiplied by three to normalize the range to the unit interval [17]. If C is close to zero, the graph contains few transitive edges; if it is close to one, such edges are the rule.

Such a quantity is of interest for two reasons. First, it is very small in random graphs, as discussed in Section 2.1, so larger values may imply more order in the processes generating the graph structure. Second, transitivity among nodes is evidence of communication in many graphs in which the nodes represent autonomous agents. As an example, if nodes a, b, and c represent users participating in a peer-to-peer file sharing network, and a shares edges with both b and c, the formation of a link between b and c suggests that a may have facilitated their communication. In contrast, users visiting a news-oriented Web site would not normally have knowledge of each other's activity, and we would expect fewer completed triangles in such a graph.

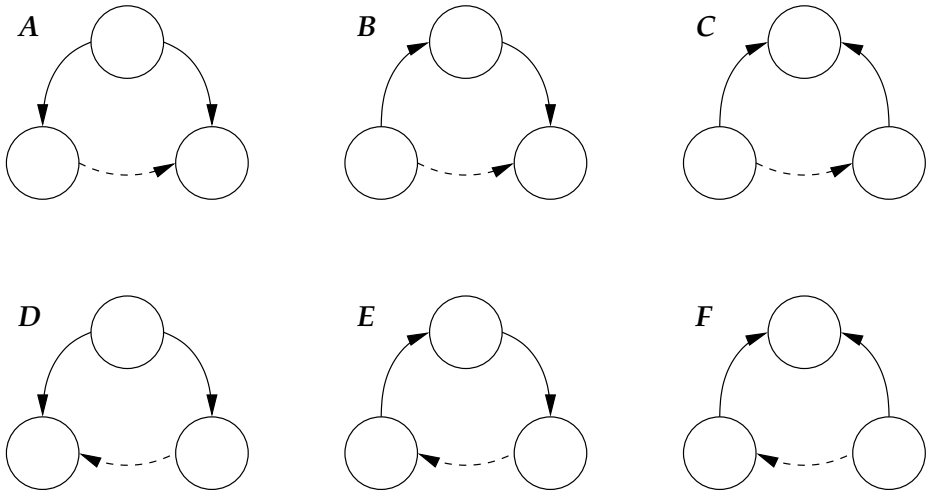


Figure 2.4: Possible node configurations for clustering in directed graphs. The addition of a third edge can produce one of two different structures, one cyclical (E) and one not (A-D, F).

Instead of regarding clustering as a property of the entire graph, we can assign a local value to each node in the graph by defining the clustering coefficient C_i of node i as the ratio of triangles to pairs of neighbors for node i [147]. If a node has a degree of less than two, we assign it a clustering coefficient of zero. The previous clustering definition is still accessible as the mean of the distribution: $C = \langle C_i \rangle$. Moreover, we gain the advantage of having a full distribution of clustering coefficients over the nodes in the graph, permitting more meaningful comparison of clustering properties across graphs. Two graphs that share the same value of C may exhibit drastically different amounts of variance in C_i across their nodes, implying different dynamics at work in their formation.

Extending the notion of clustering to directed graphs is not entirely straightforward, as shown in Figure 2.1. In this case, there are three possible configurations of three nodes for which we may be interested in the presence of additional edges. The presence of such an edge can produce either cyclic or acyclic structures, which may lead us to different normalization constants depending on which configurations are of primary interest for the graph in question.

In this work, I resolve this issue by adopting an alternate definition of the clustering coefficient C_i similar to the intuitive one presented above, but with a more direct extension to directed graphs [54]. In this scheme, we define the clustering coefficient C_i of node i as $C_i = \frac{2E(i)}{k(i)(k(i)-1)}$, where $E(i)$ is the number of edges that exist among the neighbors of node i. Informally, this is simply the number of edges among the neighbors of the node divided by the number of edges that *could* exist. This definition is then extended to directed graphs simply by defining $C_i = \frac{E(i)}{k(i)(k(i)-1)}$, where $k(i) = k_{in}(i) + k_{out}(i)$, $E(i)$ is taken over both inbound and outbound neighbors of i, and the normalization is modified to reflect the greater number of edges possible in a directed graph. This definition incorporates all three configurations found in Figure 2.1. Other definitions of $E(i)$ and $k(i)$ allow focus on a single type of configuration, but I do not consider them in this work.

Random graphs

The statistical properties of a graph are not always interesting in their own right; they require comparison to values exhibited by other graphs. In particular, it is useful to compare graphs based on observed data to those produced by various stochastic processes. Although shared properties are no guarantee that similar processes are at work in the observed data and the model, they at least hold the prospect open; moreover, wildly disparate properties are a strong suggestion that different processes are at work.

The simplest such generative process is the Erdös-Rényi random graph, which is defined by two parameters: the number of nodes N and the uniform probability p that an edge exists between any two edges selected at random [56]. We can thus expect there to be roughly pN^2 edges in such a graph. Large graphs produced with this model will have a Poissonian degree distribution rather than a heavily skewed tail [120]. They also have very low clustering coefficients and are neither assortative nor disassortative, which is to be expected, given that the presence or absence of each

edge is an independent event. However, given sufficient edge density, these random graphs do tend to contain a giant connected component and display small-world properties, so the presence of these properties does not serve as evidence of deliberate design or coordinated behavior.

Erdös-Rényi random graphs have contributed greatly to our understanding of the mathematics of graphs, but they do not resemble many graphs derived from observed data. We often observe not a Poissonian degree distribution, but rather one that obeys a power-law, so that $p(k) \sim k^{-\gamma}$. In this case, it is more appropriate to compare our observed graph not to the Erdös-Rényi model, but one that exhibits a similar degree sequence. An important property of such graphs is that we can expect the clustering coefficient to be non-zero even in the case of very large networks if $\gamma < \frac{7}{3}$, implying that we must be cautious in drawing strong conclusions about the generative process of a scale-free network based solely on its clustering behavior [8].

There are a wide variety of generative models that produce graphs with scale-free degree distributions, the most famous and widely analyzed of which is the Barabási-Albert model [16]. Often referred to the "preferential attachment" model, it rests on the assumption that when a new node is added to a graph, it is more likely to attach to a node that is already of high degree. The simplicity of such a model is compelling, but some of its features make it implausible for many situations. Primary among these is the necessity that every new node have global information on the current state of the network—an assumption that clearly cannot hold in the case of either the World Wide Web or the Internet.

There is limited utility in comparing the graphs observed in the data to those generated by either the Barabási-Albert model or any of its many competitors. Simply put, there are myriad ways to construct graphs that exhibit heavy-tailed degree distributions, so that duplication of this property means little in itself. For sources of data such as those used here, it is far more informative to construct a generative model that is plausible in the light of available domain information and

then investigate how many computable properties of the model resemble those of the observed data. An exhaustive survey of existing generative models and their relation to observed traffic data is beyond the scope of this work.

2.2 Data networks

We now shift focus from the abstract field of graph theory to the very concrete one of data networks, the Internet in particular. There are many misconceptions and seldom discussed topics regarding data networking and network architecture, many of which can lead to confusion when it comes to analysis and discussion of Internet traffic data. "Network" researchers without practical experience working with actual *data* networks can easily be sidetracked by misconceptions or drawn to features of the data that reflect the design of the network rather than the behavior it supports. This section thus contains a substantial amount of material that would seem basic to a network engineer, but is esoteric to a more general audience, and must be understood to avoid later misunderstanding.

The network stack

Fundamental to the discussion of data networking is the notion of a *network stack*, which is a framework for describing the organization of the communications protocols present in a network device. The framework most often used for such discussions is the Open Systems Interconnection (OSI) stack, shown in Figure 2.2. A network stack is comprised of *layers*—seven, in the case of the OSI model—with the most abstract layer said to be on the *top*, and the most physical layer said to be on the *bottom*.

At the very bottom of the OSI stack is the *physical layer*, which is concerned with the actual

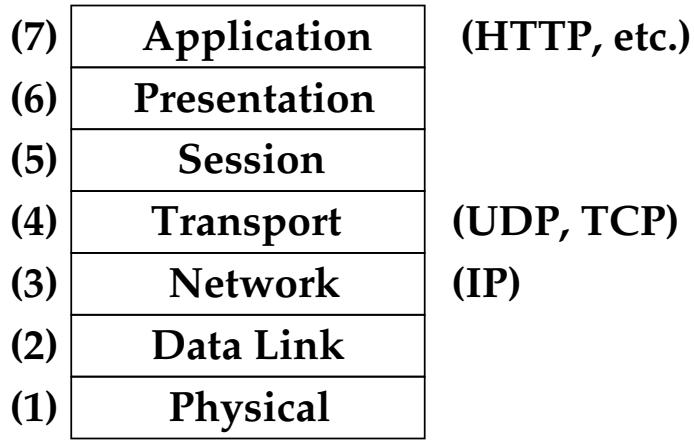


Figure 2.5: The OSI network stack. The bottom layer is the closest to the hardware, successive layers add new abstractions, and the top layer provides an interface to software applications. The core protocols defined by Internet standards are shown to the right.

physical medium that connects networked devices (e.g. copper wire or fiberoptic cable) and the signaling mechanism used to transmit bits of information. Immediately above that is the *link layer*, which provides the first significant layer of abstraction by organizing data into *frames* that contain many data bits. The link layer is the lowest layer to include a notion of an *addressing* that associates a parcel of data with a specific source or destination. Above that is the *network layer*, which introduces the notion of *routing* data to various destinations based on addressing and other criteria. Next is the *transport layer*, which adds the abstraction of multiple channels of communication between two network-connected devices. Such channels may be either message- or stream-oriented and include provisions for reliable delivery not present in lower levels of the stack. The *session layer* is based on the insight that a single transaction between network devices may involve multiple connections at the transport layers; it serves to unite these otherwise unrelated conversations. Just above that, the *presentation layer* is concerned with the interpretation of data transmitted over the network: should a particular sequence of 32 bits be interpreted as a long integer, a floating-point value, or a sequence of four ASCII characters? Finally, at the top of the stack, the *application layer* provides

the abstractions necessary for high-level applications that communicate over the network, such as fetching Web pages or transferring files.

The layers of the OSI stack are not network protocols themselves, but we can associate each networking protocol with a particular layer of the stack. This means there can be multiple protocols associated with a layer of the stack, even in a single device. For example, most Internet-connected computers support both the TCP and UDP protocols, both of which are associated with the transport layer of the OSI stack.

We have arrived at a common point of confusion: although the OSI stack was developed as a framework for an actual suite of communications protocols, those protocols were not widely adopted, and they are not the foundation of the modern Internet, whose protocols have largely supplanted them. The OSI stack is ubiquitous, but only as a theoretical framework for discussing networking protocols in general, including those used on the Internet. In many cases, this causes no great difficulty, but the mismatched origins of the OSI stack and the Internet mean that some Internet protocols do not fit neatly into the OSI framework; they may occupy several layers or further subdivide one of the layers. Furthermore, the session and presentation layers of the OSI stack have no protocols defined in the networking stacks of most Internet-connected devices. The ideas of these layers are certainly present on the Internet, e.g., PHP sessions in the case of the session layer or XML and ASN.1 in the case of the presentation layer, but these services are generally provided by individual applications and not as part of the operating system. Very few intermediate network devices have any awareness of session or presentation information, which means that most sources of network traffic data omit these layers entirely. For this reason, I follow the lead of many other Internet researchers and disregard the session and presentation layers.

Armed with this abbreviated five-layer stack, we can consider how communications actually take place between two devices on the network. An abstract view of host-to-host communication

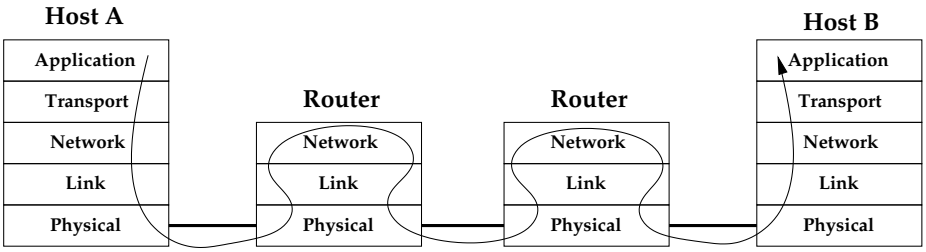


Figure 2.6: An abstract view of host-to-host communication in the Internet, showing the path of data through the protocol layers of devices in a network path.

can be seen in Figure 2.2. When a network application on Host A generates a request, the data are *pushed down* through the layers of the network stack. The data pass over a physical connection to an intermediate network device, where the data are *popped up* through several layers of the stack to allow the device to make a routing decision. The data are then pushed back down to the physical layer and transmitted to another intermediate device. Eventually the data reach the destination Host B, where the request is popped up all the way back to the application layer and delivered to an application.

All of this pushing and popping is accomplished through the use of encapsulation. Whenever a message moves from a higher layer to a lower layer, it is wrapped with additional information, usually in the form of a header or footer. This header contains a *demultiplexing key* that can be used by the lower layer on another device to select the correct higher-layer recipient of the message. This idea makes it possible for multiple protocols to occupy the same layer of the stack and for two hosts to participate in many simultaneous conversations.

Internet architecture

Having examined the basic mechanisms of network communication in the abstract, the time has come to examine specific protocols in use on the modern Internet and their implications for the

gathering and analysis of traffic data.

This task is eased somewhat by a fundamental principle of the Internet: that it truly is a "network of networks." The protocols governing the operation of the Internet apply to only the network layer and above; the Internet is almost entirely agnostic as to which technologies are used for the physical and link layer. A single path through the Internet may traverse several smaller networks, each of which use different technologies for the lower layers, using the exact same network, transport, and application protocols as if only a single link technology were in use.

Ethernet

Even though Internet protocols are unconcerned with the particular link and physical layers used in local networks, a few aspects of the link layer do affect the collection of traffic data. The most popular link layer protocols in use in local area networks today are all variations on the Ethernet protocol first developed at Xerox in the late 1970s [113]. The technical details of how various forms of Ethernet mediate access to the network are beyond the scope of this work; it is sufficient to make note of a few key details.

First, most Ethernet technologies group network data into *frames* with a maximum size of about 1,500 bytes. The technology does not guarantee the delivery of these frames to a destination host, but simply makes its best effort, leaving the detection of missing frames to higher-level protocols.

Second, each Ethernet device has a single 48-bit address, the Medium Access Control (MAC) address. In theory, this address is globally unique, associated with the hardware at the time of manufacture, and does not change. Sloppy manufacturing, malicious tinkering, and other factors can subvert these properties, but they hold true for the vast majority of all observable network traffic. This makes the MAC address the most reliable externally observable identifier of a specific network-connected device: it should never change unless the hardware changes.

The final point is related to MAC addresses. While the MAC address has great utility as a reliable identifier of a particular network host, the design of the Internet as a technology-agnostic system means these MAC addresses are only visible in the local network. If we again consider the network path shown in Figure 2.2, even if every hop on the path uses Ethernet, we will find that the MAC address of Host A is visible *only* on the link between it and the closest router. Similarly, the MAC address of Host B will be visible only on its link. Assuming that the intervening links also use Ethernet, if we were to capture their traffic, we would find not the MAC addresses of Hosts A and B, but those of the intervening routers. This means that, in the absence of multiple data collection points and the ability to correlate data between them, we cannot associate MAC addresses with both ends of any network connection that extends beyond the local network. This imposes an unavoidable lack of precision on many forms of traffic data.

Internet Protocol

The Internet has only a single protocol at the network layer, aptly named Internet Protocol (IP). There are two versions of this protocol in active use. The first, IPv4, accounts for the overwhelming majority of all Internet traffic. The second, IPv6, was developed to ease concerns about IPv4 address depletion and is supported by most major operating systems, but still accounts for only a small fraction of network activity. These two versions of IP differ substantially in their definition and operation, and many of the protocols supporting collection of traffic data about IPv4 do not yet work with IPv6, making it difficult to obtain comparable information about IPv6 traffic. Because IPv6 represents relatively little data as compared to IPv4 and there is no reason to suspect that IPv6 traffic would be meaningfully different in character from IPv4 traffic, this work omits any analysis of IPv6 traffic data. Hereafter, whenever I mention an "IP address", I am implicitly referring to an IPv4 address rather than an IPv6 address.

Every device communicating on the Internet uses IP, and every device using IP must have at least one IP address. This address is a 32-bit value, usually expressed in "dotted decimal" notion as a series of 8-bit integers separated by periods, such as 156.56.103.1. In general, these addresses are globally unique identifiers: only my workstation has the address 156.56.103.1, and no other host on the Internet can use it. This would seem to fix the number of potential hosts on the Internet at 2^{32}, or around 4.2 billion, but there are several complicating factors.

First, there are significant inefficiencies in the allocation of IP addresses. They are not assigned to institutions individually, but rather in large blocks that share the same most significant bits and differ only in the least significant bits. These address blocks are usually written using Classless Inter-Domain Routing (CIDR) notation. To represent a block in this system, we write an IP address by retaining the fixed bits ("network" part) of the block and setting the varying bits ("host" part) to zero, followed by a slash and the number of fixed bits. For example, the address block 10.199.21.16/31 contains only two addresses: 10.199.21.16 and 10.199.21.17. In the early days of the Internet, when computers were rare and expensive, institutions were given address blocks far in excess of their actual requirements. For example, the IP addresses assigned to MIT have only eight fixed bits, giving a single school over 24 million host addresses.

Second, there are a number of special blocks of IP addresses that cannot be assigned to hosts. Some are "private address blocks", such as 10.0.0.0/8, that are used only locally to an institution and cannot be routed across the global Internet. Some are multicast addresses, which are not associated with a single host, but a whole collection of hosts receiving data from a broadcast source. Another large address block, 127.0.0.0/8, is reserved for "loopback" connections whose traffic never escapes a network host. Finally, a number of addresses are reserved for experimental projects and future expansion.

All of these factors combine to reduce the number of unique, globally routable IP addresses.

The rapid expansion of the Internet in the 1990s, especially in east Asia, fueled concerns that the Internet would run out of assignable addresses and led to the proposal of IPv6. This address shortage has failed to materialize, largely due to the rise of a technology called Network Address Translation (NAT), which allows a number of networked computers to share a single globally routable IP address. A NAT system does this by dynamically rewriting IP packets: it replaces source addresses with its own address as packets leave the local network, and destination addresses with the appropriate local addresses as packets arrive from the rest of the Internet. Network engineers tend to dislike NAT, as it violates the principle of *end-to-end connectivity*, in which IP packets pass virtually unchanged between any two points on the network and no data collection point is privileged over any other. In spite of this objection, NAT technology is extremely popular and supported by millions of routers in small home networks.

The use of NAT has significant implications for the analysis of Internet traffic data: when we observe a packet with a source address of 149.159.0.27, we do not know whether this address is used by a single computer or a NAT system supporting many computers, and we have no way of finding out. What we *can* do is observe that NAT is typically used in small offices or personal residences, suggesting that the traffic of the computers behind any particular NAT system ought to be similar in character. There is no special reason for worrying that a single IP address may represent two different people using two different computers when it has always been possible for two different people to use the *same* computer.

One final aspect of IP addresses merits discussion. In Section 2.2, we saw that Ethernet addresses are associated with network hardware and fixed at the time of manufacture; the same is not true of IP addresses. The IP address of a host on the Internet can change at any time without any notice of such a change propagating through the network. This occurs infrequently for infrastructure-related systems such as Web and mail servers. However, many institutions use the

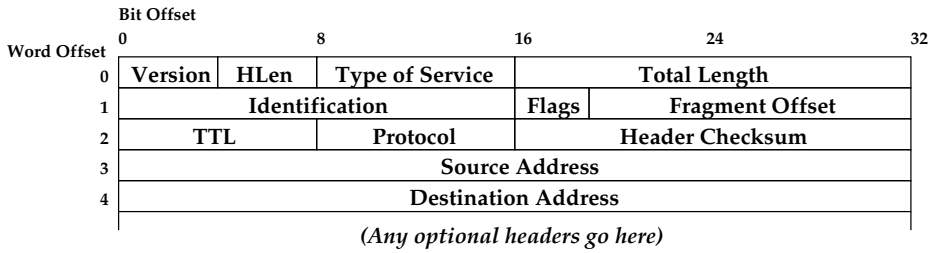


Figure 2.7: Structure of the IPv4 packet header.

Dynamic Host Configuration Protocol (DHCP) to assign IP addresses to hosts when they become active on the network, reallocating the addresses when the systems no longer require them. The address of a continuously active system rarely changes, but a period of extended inactivity often results in a host's address being assigned to another computer. The consequence is that the correspondence between an IP address and a particular computer is strong in the short term but decays over time; it is quite likely that the address 149.159.33.166 represents the same host one hour after its first observation, but much less likely three weeks later.

The header associated with IP packets contains more fields than just a source and destination address, as shown in Figure 2.2. Many of these fields are of little relevance to the present discussion, but several bear further mention. First, the *Flags* and *Fragment Offset* fields are used to support the fragmentation of a single IP packet when it must be transmitted using a link layer that cannot support frames of sufficient length. For example, a 4,096-byte IP packet must be fragmented into at least three individual frames to traverse an Ethernet link. These packets traverse the network independently followin their initial fragmentation. Second, the *Protocol* field is a demultiplexing key for the transport layer, which is described in the following section.

As was the case for Ethernet, IP is a *best-effort* protocol. It does not guarantee delivery of packets to the destination host, and it contains no provisions for automatic retransmission of lost traffic. Any formal guarantees of reliable transmission are the purview of higher-layer protocols.

Internet transport protocols

Although the Internet is built around a single network layer protocol, the 8-bit *Protocol* field in the IP header enables the use of a wide variety of transport layer protocols. In practice, however, the vast majority of Internet traffic is associated with only three transport protocols: Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), and Transmission Control Protocol (TCP). These three protocols have dramatically different purposes and designs, and they must be discussed separately.

ICMP is used for error notification, performance testing, and path tests, rather than a means of delivering application data, so one may argue that it is not a transport protocol in the strictest sense. The majority of ICMP messages transmitted across the Internet are induced by the *ping* utility: normally, whenever an Internet-connected host receives an ICMP *echo-request* message, it responds with an ICMP *echo-reply* message. The proportion of pings returned and the latency of the replies are often used as rough estimates of connection and bandwidth quality. Pings are also often used in network attacks: the perpetrators try to saturate a victim's network resources by generating a massive number of echo requests, called a *flood ping*, often from a falsified source address. A variation on this, the "Smurf" attack, targets broadcast IP addresses, which are capable of relaying incoming messages to an entire address block. Thanks to the efforts of network administrators, these attacks are much less effective than they once were, but they are often still present in network traffic data.

ICMP traffic is of only slight utility in analyzing user behavior. Much of the traffic is automatically generated, and aggregate ICMP traffic is normally low enough so that a serious throughput test or flood ping can easily contribute the lion's share of observed activity. The messages do not convey any application information and are not directly associated with other communications channels (i.e., there is no multiplexing), so the patterns of interaction are unlikely to reveal much

about user behavior.

UDP is the simplest of the transport protocols used for actual data transmission; it provides little more than a means of multiplexing message delivery between two hosts. It is a message-based protocol, with each message containing a 16-bit *source port* and a 16-bit *destination port* in addition to the source and destination IP addresses. These port values function as a demultiplexing key for application-level protocols.

There is no formal notion of a connection in UDP: each message can technically constitute a complete conversation. In practice, however, most applications built on UDP follow a simple request-reply strategy. A client system selects an arbitrary, or *ephemeral*, port number and sends a UDP message to a well-known destination port on a server. When the server replies, it repeats this arbitrary port number, allowing the client to associate this second message with the original request.

Although UDP still does not offer any guarantee of message delivery, a wide variety of application protocols use UDP as their transport mechanism. The most important of these include Domain Name Service (DNS), Trivial File Transfer Protocol (TFTP), Simple Network Management Protocol (SNMP), and the Network File System (NFS), as well as many audio and video streaming services. In addition, the commonly used diagnostic tool *traceroute* makes use of UDP by default.

While the patterns of connections evidenced by UDP-based applications are more relevant to traffic analysis than ICMP, they are still of limited utility. By far, the most commonly used UDP-based application is DNS, which would at first seem to be a fruitful source of data. However, if the *contents* of DNS messages are unavailable, there is little we can study outside of the hierarchical structure of the DNS system itself. The interconnections among DNS servers may reveal an interesting infrastructure, but they do not reflect the dynamics of user behavior.

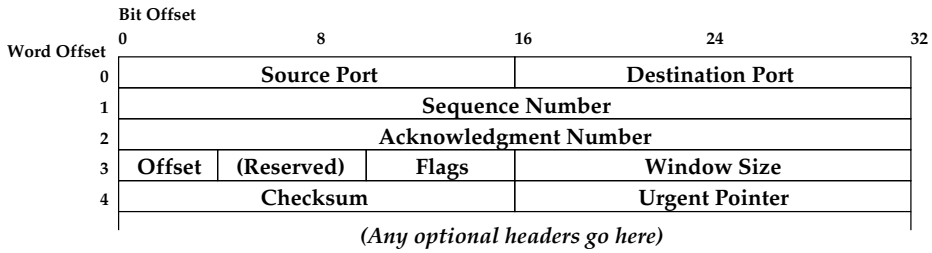


Figure 2.8: Structure of the TCP header.

The final of the three major Internet transport protocols is Transmission Control Protocol (TCP), which ferries the traffic of a large majority of network applications. In contrast to ICMP and UDP, TCP is a complex protocol with control mechanisms that have been under constant revision and refinement since its introduction in the early 1980s. Whereas UDP offers only best-effort delivery of individual messages, TCP provides reliable delivery of a stream of data, hiding the packet-oriented nature of the underlying technology from network applications. TCP also provides congestion control mechanisms and a variety of other features that help to maximize the performance of applications requiring two-way network communications.

Most of the technical means by which TCP accomplishes these tasks are beyond the scope of the present work. However, we must examine a few features of TCP in order to understand what data are represented in network flows, and how to interpret them. These features are all controlled by various fields in the TCP header, pictured in Figure 2.2.

The first salient point concerns the TCP source and destination port numbers. These port numbers have the same demultiplexing function and range as the port numbers used by UDP, but they are semantically distinct from each other: TCP port 7331 has no inherent relationship with UDP port 7331. As in the case of UDP, a client system (the host that initiates a connection) selects an arbitrary port number for its end of the connection; the server uses a "well-known" port number

that the client must know in advance. There are standard port numbers associated with all the major TCP-based applications, but these assignments are a matter of convention and courtesy rather than a technical limitation. The HTTP application protocol used by the Web will operate just as well on TCP port 7734 as it will on TCP port 80, its assigned port. A variety of factors motivate users to run server applications on non-standard ports, including the use of NATs; the desire to obscure their traffic or evade firewall restrictions; and the fact that many operating systems allow only administrative users to use ports numbered below 1024.

The next point concerns the directionality of traffic. The design of TCP means that every successful connection will generate traffic in both directions—from the server to the client and from the client to the server—even if only one party has data to transmit. One reason for this is the "three-way handshake" used to negotiate a TCP connection, in which the client and server establish initial parameters for the algorithm that guarantees reliable, in-order delivery of data. Furthermore, TCP requires active acknowledgment of received data; even in cases such as a client downloading a large file from a server, the client will generate a large number of acknowledgment messages.

Finally, we must remember that while not all TCP connections are successful, even failed connections represent actual network packets and contribute to network traffic data. In a sampled environment, it is not necessarily possible to tell which records represent successful two-way communications and which do not. Moreover, not all packets observed in real networks make any sense according to the formal definition of TCP, and attackers often have a strong motivation to emit packets that break the rules. The definition of the protocol is, in the end, simply a recommendation; the packets we see on the network are observed reality.

As mentioned earlier, nearly all network applications requiring two-way communication use TCP as a transport mechanism, and we can generally expect that any new network application will rely on TCP. There are a few exceptions to this TCP-dominated universe worth mentioning.

DNS can use either TCP or UDP, but generally uses UDP because of its lower latency and overhead for very brief communications. Streaming media applications generally use UDP rather than TCP because consistent latency is more important than reliable delivery: frames of a video that arrive late or out of order are better ignored than retransmitted. Finally, no multicast applications use TCP, since IP multicast addresses can serve only as destinations, never as sources.

Internet application protocols

There are far too many application protocols in use on the Internet for a complete catalog to be either useful or feasible. This is a basic consequence of the design of the Internet: everything above the transport layer is left entirely up to individual users of the network. Designing, implementing, and deploying a new application-layer protocol does not require modification of network infrastructure or coordination with any network authority.

The low barrier to entry for network applications associated with this open approach has been a driving force behind innovation on the Internet, but it has also spawned much of the chaos that characterizes Internet activity. Considered in isolation, no information from the IP and TCP headers of an Internet packet can give us definitive proof of what application protocol is involved. We will never have a full list of applications, and any application can be using any TCP port at any time. Indeed, an important contribution of the present work is to show that studying the *relationships* among network hosts and ports is a practical means of guessing the actual applications being used.

Although application protocols are too numerous to list in detail and their number grows continuously, we can at least divide TCP-based applications into three broad categories.

The first category is the Web, which so dominates human activity online that many use the terms “Web” and “Internet” interchangeably. All Web traffic, whether it includes an encryption layer or not, uses the HyperText Transfer Protocol (HTTP) for data exchange. HTTP is somewhat

distinct from many other application protocols not only because of the prevalence of the Web, but also because it is a "stateless" protocol, despite using TCP for transport. There is no concept of a Web session built into HTTP itself: the connection between a Web client and server lasts for only a single request unless "keep-alive" is used, which is simply an optional mechanism for avoiding the overhead of constructing and tearing down multiple TCP connections for a series of small transfers. Any state between the client and server must be handled in other ways, such as cookies, PHP session keys, URL-encoded form variables, and so forth.

The second category consists of applications that follow a traditional client-server model. In such an application, the roles of the client (the system initiating the connection) and the server (the system receiving the connection) are semantically distinct. The overall structure of the interaction generally consists of a sequence of client requests interleaved with server responses. In contrast to HTTP, many application protocols can have long-lived connections that incorporate quite a bit of state into the protocol itself. A typical example of a client-server application protocol is Network News Transfer Protocol (NNTP), the underlying technology of USENET. In NNTP, news readers (clients) contact centralized news servers to post and retrieve articles, with the protocol entering different contexts depending on the sequence of commands issued. News clients and news servers are clearly distinct pieces of software: the former is an interactive application for end users, and the latter is a data warehouse that typically runs on a dedicated system.

The third category consists of peer-to-peer applications. The protocols used by these applications generally maintain the roles of "client" and "server" only in the technical sense that a client initiates communication and a server waits for initial contact. Once a connection has been established, the distinction between these roles fades away, and both hosts are equally capable of generating or servicing requests. Nor is the contact usually limited to a single pair of hosts; peer-to-peer

applications maintain some degree of awareness of the large and complex network of interconnected hosts using the application. Whereas the Web and traditional client-server applications usually have dedicated servers that rarely, if ever, act as clients, it is common for hosts in peer-to-peer applications to both initiate and receive connections.

We must bear in mind that these categories represent a rough grouping based on different dynamics of behavior rather than a explicit statements of design. Nothing in the Internet protocol stack makes it necessary for a developer to make an explicit decision as to which category their own application will occupy. As the Internet continues to evolve, these categories may become less descriptive of the applications in wide use, necessitating the development of a new descriptive structure. An important goal of this work has been to provide an empirical basis for creating such a structure.

Internet routing

A basic understanding of Internet routing is necessary to appreciate the best vantage points for observation of traffic data from the Internet and what aspects of behavior those data can represent. We need not go so far as to examine the algorithms used by standard routing protocols; it is sufficient to discuss the basic structure of the modern Internet and how it affects the traffic visible at different points in the network.

The Internet is often described as a "network of networks," a truth which is represented at the routing level. The size and complexity of the Internet make it impossible to make routing decisions at the level of individual hosts; instead, routing policy is determined at the level of entire networks. These are not local area networks such as might exist in an individual office, but larger ones that represent the network infrastructure of an entire organization. These "autonomous systems" (ASes) are the building blocks of the Internet and represent a portion of the network under

a single administrative control. An AS can use any routing protocol internally—perhaps even an experimental one unknown to the outside world—but at the exchange points where ASes meet, the only routing protocol used is Border Gateway Protocol (BGP), which is as integral to the modern Internet as IP and TCP.

Not every organization with its own network has its own autonomous system. ASes are identified numerically in BGP and other related protocols; because the fields that hold AS numbers were fixed at sixteen bits, there can be only 65,536 distinct ASes on the Internet. This sparsity of resources means that only fairly large organizations justify having their own AS. While a research university or nationally known business probably has its own AS, a small company that simply leases a small block of addresses from their Internet Service Provider (ISP) most likely does not.

From the standpoint of BGP, all ASes are born alike. However, their functional roles in the Internet are very different. The primary distinction is between *edge networks* and *transit networks*.

As their name implies, edge networks sit on the periphery of the Internet. They are usually concentrated in a relatively small geographic location and provide last-mile connectivity for a dedicated pool of customers. The Bloomington campus of Indiana University contains a prime example of an edge AS; its network provides Internet connectivity for over 40,000 people, mostly in the city of Bloomington. The vast majority of the traffic within an edge network is directed either to or from computers inside that network, so its statistical properties are strongly influenced by the nature of its user community and the policies under which it accesses the network.

Transit networks, on the other hand, are heavily interconnected and serve primarily as couriers of information between other networks. Relatively little of the traffic found in a transit AS either originates from or is destined for a host within that network. Such a network offers a more central vantage point for observation of Internet activity. It also offers a greater diversity of users: we can regard the user population of a transit network as the union of the populations of the adjacent edge

networks. A transit network will also include users operating under differing access policies, mitigating biases introduced by firewall rules, application bans, traffic shaping, and other constraints on network access.

Network researchers face the dilemma that while transit networks are often a better source of diverse and unbiased behavioral data, their own institutions can offer access only to their own edge networks. Most transit networks are commercial enterprises operating in an intensely competitive field; the companies operating them are loathe to allow outsiders to know the physical structure of their network, let alone details of the traffic traversing them. The Abilene network operated by Internet2, described more fully in Chapter 3, is a notable exception, and Internet2 has been generous in providing access to network flow data throughout the course of this research.

2.3 Sources of behavioral data

This background in data networking can now inform a discussion of the potential sources of behavioral data from the Internet, as well as their attendant advantages and disadvantages. In keeping with the higher-level organization of this work, I first consider application-independent sources of network traffic data and then narrow my focus to available sources of Web click data.

Network traffic data

The Internet has always been comprised of a wide variety of computing devices that do not share a common hardware design, but rather a common way of interacting through standardized protocols. The descriptions of these protocols are freely available, and most devices connecting to the Internet allow their owners full access to the internal operation of the entire network stack. This open architecture has led to a wide variety of ways to obtain data on the structure and function of

the Internet, ranging from routers with dedicated data collection hardware to clusters of personal computers instrumented to provide telemetry data. Such efforts extend far beyond academia: the design and operation of traffic monitoring systems has become a vital part of the network management and security industries.

An exhaustive review of the technologies used for network measurement is beyond the scope of this work. Many of them are dedicated to analysis of the physical structure of the network or measuring the performance of various links and classes of traffic, which are unquestionably worthy endeavors, but do not produce data that directly reflects user behavior. Instead, I will focus on measurement technologies that are accessible to researchers and yield data that relates to the actual, observed behavior of Internet users. We can split these technologies into two broad categories: those based on *direct capture* of actual network packets and those based on *summary information* about packets or groups of packets.

Systems that provide direct access to network packets are quite appealing, as raw packets would seem to constitute everything there is to know about a network interaction: there can be no more complete representation of a conversation than the conversation itself. Most computers with TCP/IP stacks are able to capture packets on any of their network interfaces using either raw sockets or wrapper libraries such as *libpcap* [100] or *WinPcap* [148]. However, the placement of a system to capture packets is critical; the forwarding algorithm used by Ethernet switches normally prevents end systems from seeing traffic destined for any other host.

There are two major methods of placing a packet capture system more advantageously in the network. The first is to place it *inline* with an organization's backbone connection, so that all traffic passes through the capture system, which records packets before forwarding them toward their final destination. This is the common configuration for devices such as stateful firewalls that require access to the full volume of network traffic. Network engineers are generally wary of this approach

because the inline device creates a single point of failure in the network. If the collection system is unable to keep up with the flow of traffic or experiences a hardware malfunction, it will damage the performance of the entire network. Naturally, routers can break as well, but engineers are more tolerant toward an outage caused by the failure of critical infrastructure than one caused by a device that does not in itself provide any network services. The second method of placement involves the ability of many network switches to perform "port mirroring," which allows replicating all of the traffic on one physical port of the switch to another physical port on the switch. This allows a packet capture device to work with an identical copy of the traffic. Such a device cannot provide gatekeeper services, but the network can be indifferent to its health. This configuration will be employed later in this work, as described in Chapter 4.

One critical deficiency is common to all packet capture systems: the sheer volume of network data. A general-purpose system is simply incapable of recording to disk in real time the volume of traffic passing through a modern data network. As an example, the Internet2 network, the primary object of study in this work, allocates circuits in units of 10 Gbps [84]. A fully saturated link of this capacity carries enough data to fill almost 500 dual-layer DVDs every hour. Not only can hard drives not record information at this rate, but conventional PC hardware and operating systems are incapable of transferring packet data from kernel memory to user memory at such a pace. Any study of the full volume of packets must be done at speeds far below line rate, making real-time analysis impossible. The phenomenon of network devices that operate at speeds far beyond the capabilities of general-purpose computers is neither transient nor recent: high-performance routers are custom-built devices representing the state of the art in electronics manufacturing technology, and the tremendous economic value of the Internet guarantees that this will continue to be the case. As the speed and performance of individual PCs increases, the same technologies are applied to make the networks they run on even faster, so that line-rate analysis on modern data networks will

forever be out of reach.

There are two obvious solutions to this conundrum. The first is to sample the available stream of data, either methodically or through best-effort collection. Of these, best-effort collection is by far the easier to implement; such an approach is used for the Web click data described in Chapter 4. With this convenience does come an increased risk of sampling bias. For example, best-effort packet collection techniques may favor traffic generated at a certain time of day, or with larger or smaller packets than average. A more critical defect of sampling for content-based security applications such as Snort [91] is that sampling prevents TCP stream reassembly, making it difficult to retrieve a contiguous stretch of transmitted data from the packets or, in the case of short-lived connections, even determine whether a two-way connection was successfully established at all.

The second solution is to put aside the notion of capturing raw packets and rely instead on higher-level summaries of network transactions that describe their features without including actual payload data. The underlying premise is that such summaries can be constructed and stored using only a fraction of the computing resources necessary to cope with a full data stream. A simple example of such an approach is logging notifications from routers' access control lists: most routers can be configured to generate alert messages whenever packets matching certain criteria are encountered. However, alert generation is generally handled by the general-purpose CPU of a router, which is highly resource-constrained, making it intractable to generate alerts for the full variety of Internet traffic.

Fortunately, technical details of the design of the line cards for high-end routers have made it feasible to create another source of summary data: the network flow record. These records provide summary information for an entire network transaction without including the payloads of individual packets. There are a variety of competing formats for network flows, with each format offering a slightly different collection of fields and properties. Some formats, such as Argus [131] and Cisco's

netflow-v9 [83], allow a degree of user configuration as to what information is collected and how it is aggregated. However, the most popular network flow format, Cisco's *netflow-v5* [82], has records of both fixed length and format, making it straightforward to develop a variety of analytical tools. This format is the basis for the data collection and analysis described in Chapter 3.

The details of the *netflow-v5* format are described more fully in Chapter 3, but a few features deserve immediate comment. First, *netflow-v5* defines a network flow as being uniquely identified by a five-tuple consisting of the transport protocol used (generally UDP, TCP, or ICMP), the source IP address and transport protocol port number, and the destination IP address and transport protocol port number. These flows are unidirectional, making every successful TCP connection consist of two flows: one from the client to the server, and another from the server back to the client. Second, although UDP and ICMP are connectionless protocols, routers supporting *netflow-v5* apply a simple timeout-based heuristic to aggregate UDP and ICMP packets with identical five-tuples. Finally, even though the volume of traffic data is vastly reduced by moving from raw packets to summary information, flow information that reflects all packets from a modern, high-speed network is still beyond the capacity of routers to generate, or of workstations to analyze. Instead, routers generally synthesize flow records based on a *sampling* of the traffic they handle, which raises concerns about sampling bias in analysis conducted with this data. These concerns are also addressed in Chapter 3.

Network flows offer a tractable way to obtain a view of the full spectrum of Internet traffic across all users, protocols, and applications. However, they offer another frequently overlooked and key advantage: greater respect for the privacy of individual users. Researchers must be mindful that network conversations are *conversations* in the truest sense of the word. The packets traveling across the Internet contain not only sensitive business data and identifying information, but also personal secrets and acts of expression intended for no one but the sender and receiver. Access to raw packet data inherently raises concerns about confidentiality and trust, as well as a duty to handle the data

with extreme care and responsibility. If we can perform useful analysis without access to packet payloads, that analysis is not only more scalable, but also less prone to abuse. A primary argument of this work is that such analysis is indeed possible, to an extent not yet widely appreciated.

Unfortunately, the level of detail offered by flow data terminates at the transport layer of the protocol stack. All higher-layer details, including those of the application layer, are buried within the inaccessible packet payloads. This limits the applicability of flow data to a detailed analysis of the most popular application on the Internet: while we can observe the aggregate traffic coming to and from Web servers as identified by their IP addresses and TCP ports, we can learn nothing of the URLs actually being requested. For this information, we must turn to Web click data, as detailed in the next section.

Web click data

Because the Web is the leading Internet application and nexus of billions of dollars' worth of commerce, researchers have explored many sources of click data. Each of these sources has its own merits and faults, which I describe briefly as I build the case for my own preferred source of Web-related usage data.

The actual requests logs from Web servers provide the simplest and most obvious source of Web click data. The log entries generated by the popular Apache Web server, shown in Figure 2.3, are typical examples of such a data source. The information recorded for each page request generally consists of at least a timestamp, IP address, and requested URL. More detailed logs may include the referring URL, the Web browser used, the username provided by authenticated users, the HTTP status code generated in response to the requests, and the type and volume of data transmitted. A Web server can function as an authoritative source for all of this information, with the notable exception of the identity of the user's Web browser and the referring URL, which are derived from

```
66.249.71.38 - - [07/Mar/2010:10:44:53 -0500] "GET /~mmeiss/Nursery/data-model/0515-tree/3.txt HTTP/1.1" 200 268 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
129.79.43.211 - - [07/Mar/2010:10:44:59 -0500] "GET / HTTP/1.0" 200 - "-" "check_http/v2053 (nagios-plugins 1.4.13)"
66.249.71.38 - - [07/Mar/2010:10:46:23 -0500] "GET /~mmeiss/Nursery/data-model/0784-tree/6.txt HTTP/1.1" 200 280 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
129.79.43.211 - - [07/Mar/2010:10:54:59 -0500] "GET / HTTP/1.0" 200 - "-" "check_http/v2053 (nagios-plugins 1.4.13)"
87.250.255.241 - - [07/Mar/2010:10:55:10 -0500] "GET /~mmeiss/L579/project4/gis/world_lake.dat HTTP/1.1" 304 - "-" "Yandex/1.01.001 (compatible; Win16; I)"
67.195.112.233 - - [07/Mar/2010:10:55:45 -0500] "GET /~mmeiss/personal.html HTTP/1.0" 200 1669 "-" "Mozilla/5.0 (compatible; Yahoo! Slurp/3.0; http://help.yahoo.com/help/us/ysearch/slurp)"
67.195.112.233 - - [07/Mar/2010:10:55:45 -0500] "GET /~mmeiss/style.css HTTP/1.0" 200 1279 "http://steinbeck.ucs.indiana.edu/~mmeiss/personal.html" "Mozilla/5.0 (compatible; Yahoo! Slurp/3.0; http://help
66.249.71.38 - - [07/Mar/2010:11:01:23 -0500] "GET /~mmeiss/Nursery/data-model/0894-tree/1.txt HTTP/1.1" 200 370 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
129.79.43.211 - - [07/Mar/2010:11:04:59 -0500] "GET / HTTP/1.0" 200 - "-" "check_http/v2053 (nagios-plugins 1.4.13)"
67.195.112.233 - - [07/Mar/2010:11:12:53 -0500] "GET /~mmeiss/L597/project4/src/vmod-mean-edge-weight.c HTTP/1.0" 200 6152 "-" "Mozilla/5.0 (compatible; Yahoo! Slurp/3.0; http://help.yahoo.com/help/us/ys
129.79.43.211 - - [07/Mar/2010:11:14:59 -0500] "GET / HTTP/1.0" 200 - "-" "check_http/v2053 (nagios-plugins 1.4.13)"
66.249.71.38 - - [07/Mar/2010:11:19:21 -0500] "GET /~mmeiss/Nursery/data-empirical/0002-tree/112.txt HTTP/1.1" 200 9744 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
```

Figure 2.9: Log entries from the popular Apache Web server, illustrating the type of information available under a typical configuration.

the *User-Agent* and *Referer* fields in the HTTP header associated with the request. There are no major technical constraints on the contents of these fields; they are generally accurate only due to the lack of widespread motivation and technical savvy to falsify the information.

Server logs have the advantage that each entry is reasonably certain to reflect an actual Web request, and each request is associated with a timestamp synchronized to the same clock. Another advantage is the availability of referral data, as it provides evidence of the real-world use of directed edges in the Web graph. Unfortunately, this data reflects only how users first arrive at a site, not where they go next. An even greater disadvantage is that a single Web server has an extremely limited view of Web traffic. As we shall find later, no single server can be representative of Web servers as a whole, whether we are speaking of content, structure, or scale. Any large-scale study of traffic must therefore seek the logs of as many servers as possible, which is time-consuming and difficult to negotiate. Furthermore, because the traffic directed to Web servers obeys a scale-free distribution, a truly representative sample of server logs must necessarily include a large number of logs from low-traffic servers whose cumulative traffic contributes significantly to overall Web activity. Individually, the logs of these minor servers are not worth the time and trouble to acquire, but excluding them assumes *a priori* that browsing patterns are independent of scale.

Another source of browsing information comes from the use of "click-throughs" embedded in Web sites. In the usual scenario, a content provider has instrumented the links pointing away from their site so that a browser contacts their server again before being redirected to the external site.

A site can thus gather data on their outgoing traffic: which links were followed, by which users, and how often. When combined with server logs, this offers a fairly complete characterization of the traffic passing in and out of a particular domain. Click-through data is particularly valuable for sites whose primary purpose is to direct users to external content. For example, a search engine can determine how often users selected the first result in the list as compared to the second, and so forth. Analysis of such data has yielded useful results in areas including ranking results, page layout, and placement of links [87, 140, 68].

We must remember, however, that click-through data provides us not with a global view of Web traffic, but one limited to particular sites that have been instrumented to gather the data. Any attempt to extrapolate behavioral patterns from click-through data to the Web as a whole assumes that the sites providing the data are reflective of the Web as a whole. However, as will be explained in Chapters 3 and 4, the extreme heterogeneity found in Web traffic makes this an unreasonable assumption. Furthermore, such data is of immense commercial and competitive value, making large content providers reluctant to share such information with the research community.

A natural reaction is to instrument Web *browsers* rather than Web sites. If we gather a log of browsing history, that data will reflect the full diversity of a user's online habits without regard to the design or configuration of the sites they visit. The obvious advantages of such complete data make this mode of collection quite compelling. Indeed, it has been the basis of a number of commercial projects, such the Alexa toolbar [10].

Instrumented browsers have also been the source of ethical controversy and a justifiable target of anger from many users. As will be shown in Chapter 4, there is an astonishing degree of diversity among Web surfers, making it vital for any study involving instrumented browser data to involve as many users as possible if the results are to be reflective of the general population. This demand for a large and diverse body of data has led a number of commercial projects to abuse the notion

of informed consent, in some cases installing software on users' systems through security holes or misleading agreements. Many Web users have participated in efforts to gather browsing data with neither their knowledge nor their consent. Their privacy has been disregarded in ways leading to both emotional and financial harm. It ought to go without saying that data gathered through such unethical means is unacceptable for serious research.

The poor history (and continuing practice) of efforts to gather traffic data directly from browsers has affected the willingness of many users to participate in any such effort, regardless of the auspices under which it is conducted. This history also cannot help but exacerbate the age-old difficulty of subjects modifying their behavior while under observation; those users who are willing to participate may not browse the same sites in the same manner as they customarily do. Furthermore, developing browser telemetry that is robust, usable, and secure represents a substantial investment in time and effort, especially given the wide variety of devices now offering some form of browsing capability.

The desire to have a diversity of both users and sites represented in Web traffic data has led other researchers to analyze logs from Web proxies and caches. A variety of researchers have based their analysis of Web traffic on proxy logs [28, 47], efforts which have led to considerable overall success. This approach enjoys the considerable advantage of being unobtrusive to users, who do not need to install modified software and are not given a constant reminder of their being observed. On the other hand, the environments in which Web proxies have been deployed are often precisely those in which users must constrain their surfing habits to conform with institutional policy. The extent of this effect is debatable and difficult to quantify, but we can reasonably expect Web traffic as revealed by proxy logs to be more "work-safe" than is true in the aggregate.

There does exist one final source of Web traffic data: Web-related packets captured directly from the network. While I argued in the previous section that complete packet-level analysis of a

high-speed network connection is infeasible, such analysis is not necessary to extract useful traffic information. We require no access to the actual media files delivered to users' browsers, but only the requests. The stateless design of HTTP means that, generally speaking, the entire context for an HTTP request is brief enough to fit inside a single network packet. If just these packets can be captured at line rate, or close to it, we are left with the richest possible source of information on which URLs users are actually visiting. Moreover, the presence of the referrer field in the HTTP headers makes it possible to associate both a source and destination URL with each request, information unavailable from most other data sources.

Direct packet capture is the primary source of Web traffic information used in this work. For that reason, I defer detailed discussion of the advantages and disadvantages of this method until Chapter 4. For the time being, I will note that such data collection can offer an unobstructed view of the behavior of a large body of users and make it feasible to gather an enormous volume of traffic data in a relatively brief timespan. It also brings some attendant disadvantages, largely related to caching and redirects. Any access to packet payloads also requires careful attention to the preservation of user privacy, which has been a primary concern throughout the process of collecting and analyzing this data.

2.4 Deriving graph structures

My analysis of both network flow data and Web click data is based primarily on graph structures derived from the original time-series data. Any such derivation involves making choices about the identity of nodes in the graph, the directedness of edges, the weights associated with those edges, and the period of time over which the data are aggregated. I defer the complete technical details of how I build these graphs to Chapters 3 and 4. The goal here is simply to prime

discussion of these graphs in advance of their formal presentation by introducing their major varieties, the names I give them, and the motivation behind their constructions.

Network flow data

If we recall that every *netflow-v5* record contains information about the volume of data directed from a network source to a network destination over some period of time, it is natural to think of such a record as contributing weight to a directed edge. Each endpoint in the flow consists of a triple $(address, port, protocol)$, which we can leave distinct or aggregate as we wish. We can perform further aggregation by coalescing repeated edges during a time interval by summing their weights.

This intuition is imperfect because it assumes that the source and destination of each flow are drawn from a common pool of possible endpoints. However, we know from the previous discussion of transport-layer protocols that each host involved in a TCP flow plays a different role. Although data passes in both directions, the system initiating the connection is the client, and the system receiving that connection is the server. The asymmetric design of most application protocols makes this distinction important: clients may exhibit very different statistical properties from servers. Each flow we observe may thus describe either an edge from a client endpoint to a server endpoint, or from a server to a client. These identities are not explicitly marked in the flow records, but they can be inferred through heuristics described in Chapter 3.

The primary type of graph I use in analyzing network flow data is the *behavioral network*, by which I mean a weighted bipartite digraph in which the nodes are either clients or servers, aggregated to the level of individual hosts; and each edge represents an aggregate volume of traffic sent from one host to another. We can further refine this notion by speaking of the behavioral network for a particular application or set of applications, in which case we include only those flows with

appropriate port numbers for the server. (Recall that clients use ephemeral port numbers with no inherent meaning.) An application-specific behavioral network is thus a subset of the global behavioral network. For example, the behavioral network for NNTP would include only traffic derived from flows in which the server was associated with TCP port 119.

I will also have occasion to discuss an *application network*. In this network, the nodes represent not hosts, but network applications as identified through their customary TCP server port numbers. This graph is undirected and fully connected, with the weight between two nodes reflecting some measure of similarity between those nodes. I describe the means by which such a similarity can be reckoned in Chapter 3.

Web click data

Web click data derived from capture of HTTP headers contains information that can be used to build a variety of graph structures. For each request, we have access to a timestamp, the IP (and possibly MAC) address of the client system, the IP address of the Web server, the full URL of the resource being requested, and the referrer URL.

If we associate nodes with the client and server IP addresses and aggregate edges based on repeated requests, we obtain a network much like the behavioral network for the Web, except that all edges are from a client to a server and the weights represent the number of requests rather the volume of traffic. Such a network has utility for confirming the results of analysis done on the Web behavioral network, but this choice of endpoints does not help us in understanding Web traffic as an activity that takes place on the actual substrate of the Web. When we discuss the structure of the Web, we generally do not consider clients to be part of the Web at all: what matters are Web pages and how they are linked together.

We can place Web traffic in the context of the physical Web by instead identifying nodes with the referrer and destination URLs found in each request. When viewed in this way, each HTTP request represents the traversal of an actual link from the real Web graph. A number of caveats do apply to this interpretation. First, not every request includes referrer information. If a user opens a bookmark, returns to their browser start page, types in a URL explicitly, opens a link from an external application, or purposefully blocks this information, the associated request will not contain any referrer data. Second, referrer information can be spoofed by the browser; we have no guarantee that the outgoing link really exists. Finally, not every request is successful; without access to HTTP response codes, we cannot be certain if the resource requested actually exists or was delivered. These concerns are discussed further in Chapter 4.

If we build a graph structure by coalescing these referrer-to-destination edges over a period of time, we obtain the *Web traffic network*. Subject to the qualifications already mentioned, this network is a proper subset of the actual Web graph in terms of its nodes and edges. Its weights include additional information in the form of the number of times we have actually observed a Web user traversing each link.

We can examine the Web traffic network at different scales by applying various levels of aggregation to its nodes. We have already examined the case in which no aggregation is performed, which I will occasionally refer to as the *page traffic network*. Because the number of distinct URLs represented in the data rises so quickly as a function of time, it can be more practical to aggregate nodes based on the host names associated with the URLs, which yields the *host traffic network*. This level of aggregation can be sensitive to the design structure of major Web sites: some present a single host name to the world (e.g., Google), and others redirect traffic from a large collection of independently named servers (e.g., Facebook). We can mitigate this effect by aggregating based on the "second-level domain" (SLD) of the host names, yielding the *SLD traffic network*. For example,

URLs from *www1.foo.com* and *www2.foo.com* would be distinct in the host traffic network, but part of the *foo.com* node in the SLD traffic network.

It is also worth noting that we must derive the host name for the destination based on the virtual host name found in the HTTP header rather than the destination IP address of the captured packet. Virtual hosting and round-robin DNS queries mean that there is a many-to-many relationship between IP addresses of Web servers and the actual sites being hosted. Inclusion of the virtual host field became mandatory under version 1.0 of HTTP, making this information almost universally available in actual request packets. Because the virtual host is application-layer data, this information is entirely absent from network flow data.

2.5 Analytical techniques

The graph structures described in the previous sections are often extremely large, on the order of 10^6 to 10^8 nodes. The immense scale of the data prevents direct visualization of the graphs: not only are graph layout algorithms prohibitively expensive to compute even for much smaller graphs, but current display technology cannot devote even a single pixel to each node in a graph representing 30 million distinct network hosts. Not only direct visualization but also many standard algorithms for graph analysis are intractable for structures of this size and complexity. This is especially true when the storage requirements of their connectivity matrices exceed main memory capacity for a modern workstation. Many of the available packages for the analysis of large graphs draw the line several orders of magnitude lower or require that a representation of the graph at least fit within physical memory.

These difficulties and limitations have motivated the development of a significant quantity of

custom analytical software over the course of the research described in this dissertation, which I describe in detail in Appendix A. I cannot claim that these tools continue to represent the state of the art in the analysis of large graphs; some software solutions now exist that were either unavailable or much less powerful at the onset of this research. My guiding principle has been that algorithms of $O(n^2)$ or higher are either intractable or unreasonably complex for graphs on the order of 10^7 nodes, but that algorithms of $O(n \log n)$ are usable with careful attention to memory management.

It is tempting to argue that sampling of these large graphs can make them more amenable to complex graph analysis algorithms, but this is problematic for several reasons. It is unclear exactly how one ought to define sampling on a large graph. Do we first select nodes, and then take the edges that happen to be incident to them? Or do we first select edges, and then take the adjacent nodes? According to what distribution do we select these nodes or edges: uniformly at random, or based on the actual distributions from the graph? The literature on sampling large graph structures is fairly sparse and tends to focus on how a particular form of sampling does or does not preserve some property of interest. These results are generally analytical and assume some known generative model for the graph being sampled, leaving no clear path of action when the very purpose of the analysis is to find whether such models adequately describe the data. In general, we find that different forms of sampling are appropriate for preserving different properties of the underlying graph, and that this relationship can be highly dependent on the (still unknown) basic structure of the graph itself. These difficulties seem unlikely to persist forever, but my own lack of expertise in sampling statistics and the relatively short time for which researchers have been working with empirical data sets of this magnitude have cautioned me against any great reliance on sampled graph structures in this analysis.

Fortunately, the graph measures described in Section 2.1 all prove to be tractable for the available data sets. Distributions of degree, strength, and weight, and the scaling relations among them,

as well as measures of clustering and assortativity, can all be recovered even from very large graphs. However, great care must be taken in characterizing these distributions: as we shall see in Chapters 3 and 4, they often bear no resemblance to either a normal or Poisson distribution, exhibiting extremely long and heavy tails over several orders of magnitude. In the following section, I describe the analytical techniques used in this work to measure and characterize heavy-tailed distributions.

Heavy-tailed distributions

A natural way to visualize any distribution in a way that allows us to guess at its basic form is the *probability density function* (PDF). The PDF $P(x)$ of a random variable x obeys the property that $P(x) \geq 0$ always, with $\int_a^b P(x)\,dx$ giving the probability of a value from the distribution being in the interval $[a, b]$. In other words, the likelihood of a measurement being in the range $[a, b]$ is given by the area under the PDF between a and b, as illustrated in Figure 2.5. A basic consequence of this definition is that $\int_{-\infty}^{+\infty} P(x)\,dx = 1$, which is simply equivalent to the statement that the cumulative probability of all possible outcomes is equal to one. It is therefore straightforward to approximate the PDF for a distribution by calculating a histogram over the available measurements and normalizing the bin totals by their width and the total number of samples.

For normally distributed data, we can graph the estimated PDF on linear axes, with the expectation that as the number of available measurements increases, we can increase the number of histogram bins and come to approximate ever more closely the familiar bell-shaped curve. This does not hold for heavy-tailed distributions, where the vast width of the tail squeezes most of the data into an extremely narrow area at the left-hand side of the graph, as shown in Figure 2.5A. Furthermore, the probability density is still quite low in the tail—it is the *total* area under the tail that represents a significant proportion of the distribution, not any small segment of it. This makes the

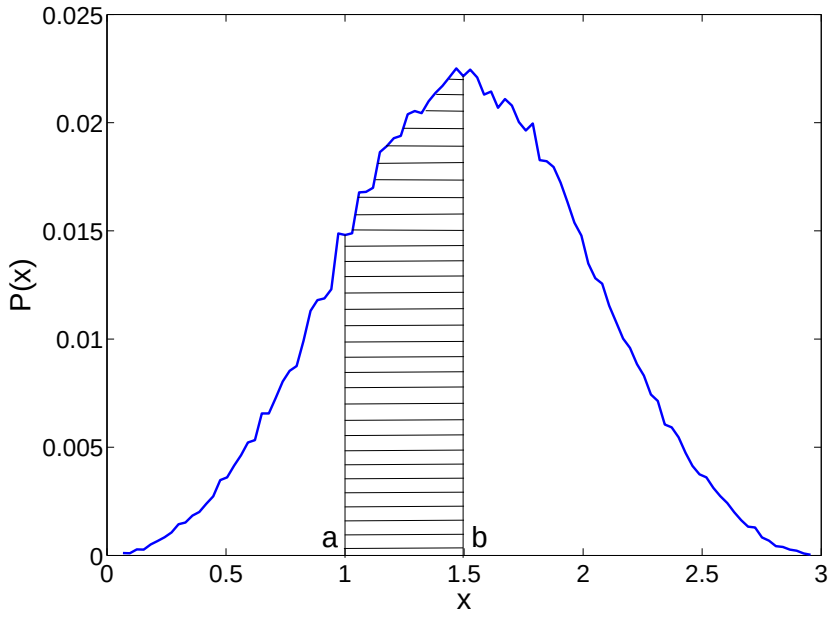


Figure 2.10: Example of a probability density function (PDF). The area under the entire curve is equal to one. The area under the curve between points a and b is an estimate of the probability that a measurement will be drawn from the interval $[a, b]$.

vertical range of the graph similarly difficult to interpret: only the left-hand side appears to have any measurable likelihood, despite the actual measurements forming the tail.

An understandable reaction to such a situation is to present the PDF of a heavy-tailed distribution on dual logarithmic axes, as shown in Figure 2.5B, which depicts the same distribution as in Figure 2.5A, save a shift to log-scale axes. When presented in this way, the PDF of a heavy-tailed distribution often appears to be roughly linear, leading us to suspect that the distribution may obey a power law $P(x) = \beta x^{-\alpha}$, where α is the absolute value of the slope of the line best fitting the tail.

It is here that we must observe the greatest caution. If we maintain equally-sized bins, we obtain a cloud of sparsely populated histogram bins at the end of the tail, most of which contain only a few measurements, making it difficult to estimate the slope of the power law with any degree of accuracy. As we shall see, obtaining an accurate estimate of the slope is of key importance: it can determine, for example, whether the variance of the distribution is bounded or not.

Furthermore, as a variety of statisticians have cautioned, fitting to this estimate of the PDF can lead us to mistake a distribution as obeying a power law when it is better characterized as a log-normal [43, 38]. In a log-normal distribution, the logarithm of the random variable is normally distributed, resulting in a much different form for the PDF:

$$P(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left[-\frac{(\ln x - \mu)^2}{2\sigma^2}\right]$$

where μ and σ are the mean and standard deviation of the logarithm of the random variable rather than its raw values. While such a distribution can indeed be long-tailed, it differs from a power law in several vital respects. While its variance is large with respect to the normal distribution, that variance is bounded, and the distribution does display a central tendency. A graph with a log-normal distribution of degree does not share the scale-free properties associated with a graph

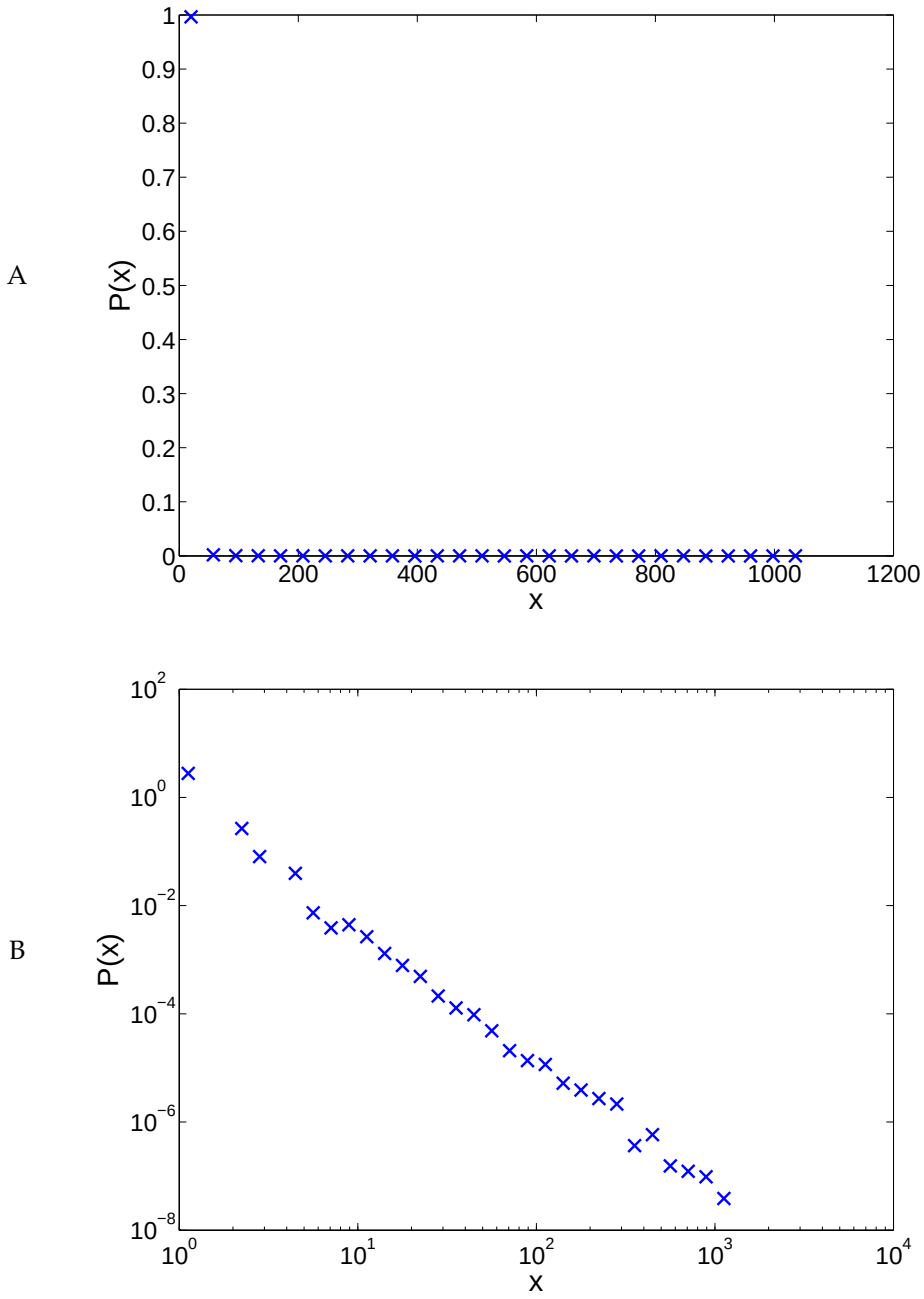


Figure 2.11: Views of the same PDF, plotted on linear axes (A) and dual logarithmic axes (B).

with a power-law degree distribution.

The desire to characterize the distribution properly and obtain an accurate measurement of its slope, should it turn out to be a power law, motivates a variety of alternative approaches, as shown in Figure 2.5. All three parts of the illustration depict the same underlying distribution, using different techniques to display and characterize its properties.

In Figure 2.5A, we see the result of fitting a slope to the tail using least-squares approximation after applying a logarithmic transformation to both axes. As mentioned above, this technique suffers from a variety of critical difficulties, and I have eschewed it whenever possible in this work.

The desire to "clean up the tail" naturally suggests the approach of using histogram bins that are equally sized in logarithmic space. This requires the additional step of normalizing each bin individually by its untransformed width as well as the size of the distribution, but it results in a much cleaner plot, as shown in Figure 2.5B. We can thus obtain a more satisfactory least-squares approximation to the slope of the tail. However, this technique tends to "over-clean" the data, which can again lead us to diagnose a power-law when a log-normal fit would be more appropriate, or to measure the slope inaccurately.

One recommended solution is to characterize the distribution based not on the PDF, but rather on the *cumulative distribution function* (CDF) [52, 53]. For any value x, the CDF $C(x)$ gives the probability that the next value of the random variable will be less than or equal to x. This is thus a monotonically non-decreasing function, with $C(x) \to 0$ as $x \to -\infty$ and $C(x) \to 1$ as $x \to +\infty$. It is often more convenient to work with the *complementary cumulative distribution function* (CCDF) rather than the CDF, which is simply defined as $\overline{C}(x) = 1 - C(x)$. We can also see that the CDF $C(x)$ of a distribution is directly related to the PDF $P(x)$:

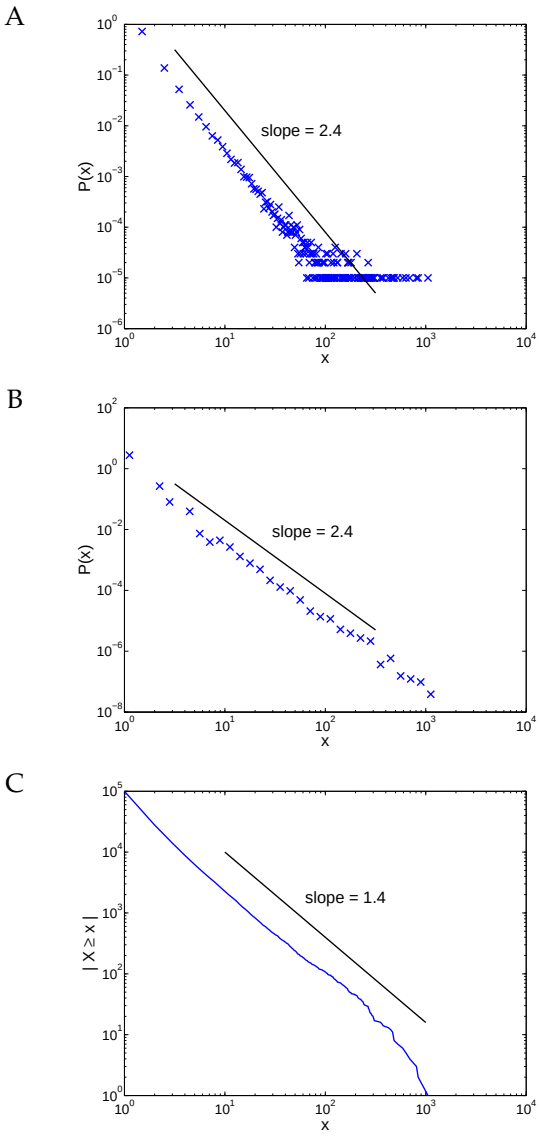


Figure 2.12: Three methods of estimating a power-law fit. The data are 100,000 points randomly sampled from a power-law distribution $P(x) \sim x^{-2.4}$. A: Least-squares approximation to histogram. B: Least-squares approximation to log-binned histogram. C: Complementary Cumulative Distribution Function (CCDF).

$$C(x) = \int_{-\infty}^{x} P(t)\, dt$$

This relationship is of great interest. Since a power-law distribution has the form $P(x) = \beta x^{\alpha}$, it therefore follows that

$$C(x) = \int_{-\infty}^{x} \beta t^{\alpha}\, dt = \frac{-\beta}{\alpha + 1}\, x^{-(\alpha+1)}$$

This means that if we plot the CCDF of a distribution on dual logarithmic axes, we can estimate the absolute value of slope of the tail as being equal to $\alpha+1$, allowing us to characterize the exponent of the power law. This is graphically depicted in Figure 2.5C, in which we again employ a basic least-squares fit to obtain an estimate of α. This technique has the advantage of representing every data point without any potentially misleading binning. It can also offer us greater confidence that we are working with a power law rather than a log-normal distribution: some distributions that appear linear when plotted on dual-log axes with logarithmic histogram bins reveal a curved shape when the CCDF is shown instead [53].

In this work, I have estimated the slopes of power-law distributions using both the PDF as approximated with a logarithmically binned histogram and the CCDF, taking the CCDF as the more reliable guide. However, I customarily show the PDF rather than the CCDF in figures, as I find it to depict the underlying data in a more intuitive manner. Most figures thus show a PDF with an estimate of the slope drawn from analysis of the CCDF.

More recently, Clauset *et al.* have recommended a technique for determining the exponent of power-law data using a maximum likelihood estimator in combination with a goodness-of-fit

statistic. [43]. The methodology they present has the advantage of providing a well-founded estimate of uncertainty for the exponent. It also suggests an optimal "cut point" in the distribution, beyond which we assume ourselves to be in the tail.

These techniques are just beginning to attract wider attention in the research community and are unfortunately much more computationally expensive than the other methods presented. As noted when appropriate in Chapters 3 and 4, I have applied them to a selection of relevant distributions and found their results consistent with my existing estimates based on analysis of the CCDFs. Indeed, the sheer quantity of data available to me seems to make all available techniques fairly robust indicators of the true slope of the distribution under consideration.

There are a number of cases, particularly in Chapter 4, in which a distribution is best described by a log-normal. In each such instance, I use maximum likelihood estimation to fit the parameters of the distribution.

The difficulties inherent in calculating the parameters of power-law distributions also apply when attempting to characterize the relationship between two distributions of quantized data, each of which is also heavy-tailed. The natural approach of constructing a two-dimensional histogram to approximate the joint distribution exacerbates the problems encountered when approximating the PDF of a single random variable. A linear scale turns out to be inappropriate for both axes, and we are drawn to using logarithmically sized histogram bins normalized by the area of the bin, with all the attendant hazards from the previous situation.

There is an additional problem in trying to visualize a two-dimensional histogram of the joint probability of heavy-tailed distributions. If the two variables are highly correlated, applying a single logarithmic transformation to the bins (as an index into a color scale, for example) may be insufficient to make much of the data visible. Put another way, if the random variables X and Y both obey power laws, it can easily be the case that the distribution of values from X that correspond to

a single value of Y is itself heavy-tailed, and vice versa.

There are several possible approaches to this problem. One is to apply a double logarithmic transformation to the data: we simply have the visualization depict $\log\log P(X,Y)$ instead of $\log P(X,Y)$. This turns out to be too strong a solution in practice and can easily lead to misleading interpretations of the data. A more appropriate approach, which I have adopted in this work, is to prepare a two-dimensional plot of the joint distribution by calculating another normalized log-binned histogram for the values of Y corresponding to each value of X. In other words, we compute independent histograms for $P(Y|X = 1)$, $P(Y|X = 2)$, and so forth, and then depict these histograms side by side on dual log axes. An example of such a plot can be seen in Figure 3.9.

The approach described above also suggests an approach for numeric characterization of the relationship between the variables. If we assume that this relationship takes the form $y = \beta x^{\alpha}$, then we can find the mean of each of the histograms and perform a simple least-squares fit in logarithmic space to determine the value of α. This method does have disadvantages (beyond its requiring this much text to explain). As mentioned above, the distributions $Y|X = 1, ...$ can themselves be heavy-tailed and exhibit extremely large variance. We can avoid leading ourselves astray by graphically depicting these means together with error bars scaled to the standard error of the mean. Another possible solution, not explored here, would be to apply the same technique to the transpose of the relationship and examine X as a function of Y. If we assume that $x = \beta' y^{\alpha'}$ and verify that $\alpha'^{-1} \approx \alpha$, we can be more assured of the analysis.

Other analytic techniques

There are a few other techniques related to the analysis of complex networks that I explore in this work and which deserve introduction here. These approaches are applicable to a wide variety of graph structures, even if I have not applied them universally, and their further investigation may

be fertile ground for future research.

We may know that a graph exhibits heavy-tailed distributions in its distributions of degree, strength, and weight without having a clear sense of whether the extreme diversity among weights applies only at the global level or extends to the level of individual nodes. We may find that the edges incident to a node of high degree tend to have roughly equal weights, or we may find that they are wildly different from one another. We can quantify the heterogeneity of weights at the level of individual nodes using a measure of *disparity*. In particular, I use the Herfindahl-Hirschman index, which the field of complex networks research has borrowed from economics [77, 78, 18, 11]. The Herfindahl-Hirschman index Y of node i in a graph is defined as

$$Y_i = \sum_j \left(\frac{w_{ij}}{s(i)} \right)^2$$

where w_{ij} is the weight of the edge connecting nodes i and j and $s(i)$ is the total strength of node i. This measure can easily be extended to directed graphs by substituting either w_{ji} and $s_{in}(i)$ or w_{ij} and $s_{out}(i)$ for w_{ij} and $s(i)$.

The Herfindahl-Hirschman index will be at its minimum when the strength of a node is evenly distributed among all its edges. In this case, $w_{ij} = s(i)/k(i)$, so $Y_i = \sum_j (1/k_i^2) = 1/k(i)$. The measure is at its maximum when the strength of a node is concentrated at a single edge, in which case we have some $w_{ij} \approx s(i)$, so $Y_i = 1$.

This disparity measure will be revisited in Chapter 4 when we evaluate whether the assumptions made by the random walker model of PageRank are supported by empirical observations of Web traffic. In that application, the edges will represent Web links weighted by user traffic; however, the index is in theory just as applicable to behavioral networks derived from flow data.

Another avenue of attack in attempting to understand the structure of complex networks involves *spectral analysis*, which refers to examining the properties of the principal eigenvectors and eigenvalues of either the boolean or weighted adjacency matrix of a graph. Such an endeavor may seem hopeless for graphs of the size under discussion, since finding the eigenvectors of an $N \times N$ matrix is $O(N^3)$ in the general case. However, the situation is not so grim as it seems: we are generally interested in only the first few eigenvectors, and there exist algorithms that perform astonishingly well on extremely sparse matrices of the type under consideration.

The motivation for examining the principal eigenvectors is to provide a compact representation of typical behavior in the network and to assist in identifying anomalous nodes. This approach to anomaly identification in network traffic was first proposed by Lakhina *et al*., who reported success in identifying otherwise inobtrusive behavior by applying thresholds to entries in the fourth eigenvector of the weighted adjacency matrix. In that case, the matrix involved was quite small, containing only the core routers of the Internet2 network. As shall be seen in Chapter 3, the results are more difficult to interpret when applied to host-level representations of traffic flow.

A final technique to mention is that of k-core analysis. The principle behind k-cores is simple: to find the k-core of a graph, we simply recursively remove all nodes with degree less than k. The fact that this removal is performed recursively means that in the resulting sub-network, every node has a degree of at least k, implying that every connected component of the graph must have at least k nodes (or $k + 1$, if self-loops are not permitted). Intuitively, the nodes belonging to the highest-numbered k-cores are extremely well-connected and play a central role in processes taking place on the network. Moreover, the k-cores of a graph are far less computationally difficult to calculate than many measures of centrality. These properties have led to the promotion of k-core analysis as the basis for visualizing extremely large networks in an efficient, meaningful, and reproducible way [12, 20, 143].

There is a slightly different impetus for the use of k-core analysis in this work. As we travel to deeper and deeper cores of a graph (i.e.., k gets larger), the number of remaining nodes $N(k)$ shrinks. Similarly, the number of connected components $C(k)$ may either increase or decrease at each step, though it must dwindle to a single component in the long run. The relationship between k, $N(k)$, and $C(k)$ may yield insight into the structure of a graph. Even if we are unable to determine the generative process underlying a graph based on this data, we can still apply it as a "fingerprint" to provide a rough measure of similarity between graphs of different size and origin. This idea will be explored in the context of behavioral networks in Chapter 3, but could also be applied to subset of the Web graph.

Note that the idea of isolating cores of a graph based on degree can easily be extended to strength. In this case, we simply define the s-core of a graph as being the subgraph obtained by recursively removing all nodes with a strength less than s. This idea has not been widely explored in the literature and is discussed in conjunction with k-core analysis in Chapter 3.

2.6 Related work

In this section, I touch upon some projects and research in related areas, especially the growth and evolution of the Internet and Web, characterization of network traffic based on flow data and other high-level descriptions, and analysis of Web usage. For each project, I describe its relation to the present work and how its direction and scope differ from my own efforts.

Network flow research

There is a large and ever-growing body of research into the growth and structure of the Internet, most of it focused on the physical Internet as defined by the connections among routers and ASes.

The decentralized nature of the Internet prevents the existence of any privileged viewing position that allows researchers to view the entire system, so they must generally rely on distributed measurements and a variety of heuristic techniques to assemble local views in a "best-guess" overview of the global network. In some cases, these local views come from passive measurement techniques such as analysis of routing tables; in others, they derive from active probes made by tools such as *traceroute* or custom software.

The most prominent Internet mapping projects [122, 32, 50] have yielded views of the Internet comprehensive enough to aid researchers in discovering remarkable properties of Internet topology, especially extreme heterogeneity in the degree distributions of the graphs of router and AS-level interconnections [62, 31, 38, 126]. Claims of power-law distributions in the Internet topology have ignited some debate as to proper methodology for characterizing long-tailed distributions (helping to inspire the emphasis on technique in the present work) and whether some results may simply reflect sampling bias associated with distributed topology measurements [37, 93, 2]. These controversies notwithstanding, more recent results do suggest that while some sampling bias is inevitable, extreme heterogeneity in degree distributions is indeed a genuine feature of the Internet [46, 135]. These findings have inspired significant activity in modeling the development of the physical Internet, with particular focus on the dynamics of its growth [104, 86, 150, 61].

These efforts have contributed greatly to our understanding of the physical structure of the Internet at several levels of detail. However, the physical network is only the substrate on which actual user-to-user interactions take place. While the structures of behavioral and application networks are undeniably influenced by the physical network, the end-to-end design philosophy of TCP/IP makes this correlation between virtual and physical topology an optimization rather than a requirement of network applications. We cannot understand the structure and dynamics of network traffic by studying its streets and thoroughfares in isolation; we must gather actual behavioral

data.

A number of projects have sought to characterize the behavior of various network applications through analysis of access logs and proxy servers [39, 1, 13, 75]. Such efforts yield insight into individual applications on a case-by-case basis, giving us typical rates of usage, a rough fingerprint of network activity, and other related information. What they fail to offer us is a global perspective that regards all network applications in the same light and studies their overall patterns of activity and how they differ from one another.

Network flow data, particularly from a transit AS, is one of the few available sources of traffic data that can provide such a wide view of user activity. The present work is far from the first to focus on network flows as a primary source for Internet behavior data. For example, inter-domain traffic has been studied on a global level through analysis of aggregate traffic exchanged by specific service providers [144]. The CAIDA measurement infrastructure has used a similar strategy involving the construction of traffic matrices at the AS level [42, 81]. A number of flow analysis projects have been conducted in response to increasing focus on the evaluation of Internet performance on a global level [35]. Projects such as the *PingER* monitoring infrastructure [137] and RIPE [60] have conducted active collection of flow-based performance data among a large number of source-destination pairs.

The attention to network flow data is not purely academic. Network engineers and security professionals increasingly view flow data as a vital resource for monitoring and security, and a variety of tools have been developed to analyze *netflow-v5* records. Mark Fullmer's *flow-tools* [67], CAIDA's *cflowd* [33], *FlowScan* [34], and CERT's *SiLK* [142] have all been widely recognized and used in capacity planning, bandwidth management, and intrusion detection, as well as basic academic research. In addition, commercial security systems such as Arbor Networks' *Peakflow* [119] use network flow data for statistical trending and anomaly detection. The *Autofocus* project uses

flow data for automated discovery of dominant and unusual traffic clusters, subject to a limited set of criteria [59].

These tools, and most of the research they support, take a relational view of network flow data. In other words, each record is regarded as a row in a database table, which can then be subjected to a variety of standard data mining techniques. They are well-suited for examining properties such as the proportion of traffic generated by particular applications or the longevity of certain classes of connection. The approach espoused in this work is fundamentally different: I regard a flow not as an isolated tuple, but as a directed edge in a graph. My attitude is not unique; a variety of recent projects share this basic approach to flow analysis, as described below.

As mentioned in the previous discussion of spectral analysis, good results in detecting traffic anomalies have been obtained through principal component analysis (PCA) of flow-based time series data [94, 95]. This approach resembles the present work in that it considers network flows as contributing weight to the router graph of the network. On the other hand, it focuses on network infrastructure rather than users and presumes the existence of "typical" traffic patterns along subspaces of maximal data variance. As we shall see in Chapter 3, this latter assumption may be unrealistic, since real-world network data can exhibit unbounded variance under normal conditions.

A more recent project used manifold learning algorithms instead of PCA to create a system that reduces the dimensionality of flow data for visualization [128]. That system allows the user to explore the relationships among a variety of entities represented in flow data, including TCP destination ports. The analysis of application networks described in Chapter 3 also regards ports as independent entities, but takes the additional step of clustering them hierarchically and using this hierarchy to predict the function of unknown applications.

There have been some other applications of complex systems analysis to the graphs formed by

network applications, especially that of the Web. Indeed, some of these projects have been a direct inspiration for the present work. I defer a discussion of relevant Web research as being more appropriate to the next section. However, other overlay networks have been examined in similar fashion, most notably email interaction and peer-to-peer networks [54, 121, 133, 132]. Researchers have also analyzed overlay networks for security purposes, focusing on monitoring and characterization of the spread of computer viruses on the Internet [116, 138, 127, 63, 121] and other malicious activities [117, 69, 152, 136].

The present work is distinguished from the majority of these efforts in two key respects. First, my primary interest has been in characterizing the behavior of *users*, taking into account the full breadth of their traffic across all network applications. To the extent that I have focused on individual applications, it has been to characterize them in a way that makes it possible to compare them with one another in a quantitative way. Assembling a view of the overlay network of a particular application through active measurement or using technical details of its protocol, besides requiring access to raw packet data, would necessitate comparing graph structures that have been assembled through very different means. Unless our knowledge of each behavioral network is gathered in the same way, we cannot be sure whether the results of comparing these networks are simply reflections of the biases inherent in the measurement techniques employed. Second, much of the existing research has been focused on the structure of overlay networks (e.g., degree) to the exclusion of the activity that takes place on them (e.g., strength). This represents not an oversight so much as a consequence of the fact many projects have used data sources that cannot provide reliable data on the intensity of interactions in a behavioral network.

There is another reason to favor an application-agnostic approach to studying traffic. A great many studies have examined the structure of the overlay networks associated with individual peer-to-peer applications (e.g., [132, 99]), but the popularity of any particular peer-to-peer service has

proven to be fleeting at best: Napster has all but vanished, few even remember WinMX or Hotline, and Gnutella and eDonkey are fading away as users continue to gravitate toward Bittorrent. I believe it appropriate to avoid consideration of any particular P2P application in favor of treating them as a general class whenever possible. While this does make my conclusions less specific in some respects, it is faithful to the precept that behavior is best characterized by the *effect* that flows have rather than their technical identity. Indeed, the results described in Chapter 3 imply that P2P applications affect the network in similar ways independently of the specific protocols they use.

A number of projects have applied machine learning techniques to flow data for traffic classification tasks. For instance, the BLINC system uses flow data to develop "graphlets" that describe the normal usage patterns of a variety of network applications; these structures are then used in conjunction with host information to predict the application associated with a given flow without regard to the port numbers used [88]. My own effort is focused on classifying applications themselves rather than their individual flows. Furthermore, my objective has been to develop this classification based entirely on observed behavior, without prior knowledge of the design of the application protocols concerned.

Other projects have used Bayesian analysis [115] and unsupervised clustering algorithms [57] to associate flows with applications; the latter has even been extended to work in circumstances where only asymmetric flow data is available [58]. I believe that my own approach could be combined with these techniques to produce a robust hybrid system for traffic classification. Such an effort represents a promising direction for future research, but is beyond the scope of this work.

A few efforts have applied machine learning and clustering techniques to partial packet traces in order to classify flows in real time [22, 151]. These systems are quite promising for security applications: they not only enjoy the scalability of flow data, but also make classification possible even while a flow is still active. Unfortunately, their requirement for actual packet data puts them

firmly in the domain of organizational security. The goal of the present research has been to show that it is possible to characterize and model the activity of a large population while maintaining the privacy of those users.

There have been some relevant efforts to use behavioral data other than network flows for anomaly detection. One such project involved applying machine learning techniques to identify anomalous application behavior in the context of a single host [70]. A similar effort implemented an intrusion detection system by distributing intelligent agents across many hosts and correlating their results [76]. Both of these efforts were essentially host-based rather than network-based, making them of greater benefit to the administrators of individual hosts than to researchers interested in the collective behavior of many users.

Finally, researchers have explored using unsupervised clustering techniques as a way of applying summarized network traffic data to intrusion detection [129]. Leung *et al.* have described techniques for applying other unsupervised machine learning methods to summarized traffic data in order to identify anomalies without having to model normal traffic levels [98]. However, my very purpose is to characterize normal traffic levels, to the extent to which such a characterization proves possible. Furthermore, the efforts just mentioned do not consider the structure of behavioral networks derived from the flow data. The present work aims to provide a better understanding of the topology of behavioral networks, which may improve the performance of these security applications.

Web click research

The World Wide Web is of course a major subject of study, with a number of high-profile conferences aimed exclusively at Web research. There exists a large and growing body of research into the structure of the Web graph, the dynamics of its evolution, and the ways in which users navigate

and interact with it. My intent is not to outline this entire field of research, but to draw attention to efforts of particular relevance to the present work. Some have helped to inspire this research; others have shared similar goals while taking much different approaches.

Many Web mining studies focus on the static network represented by the link graph, in which vertices and directed edges identify Web pages and hyperlinks, and links are seen as endorsements among pages. Data gathered in large-scale crawls [15, 30, 92, 4, 96] of the Web have shown the Web to exhibit a rich and complex architecture. Among the important insights into the large-scale structure of the Web graph given to us by these crawl-based studies are the "bow-tie" model, the presence of self-similar structures and scale-free distributions, and the Web's small-world topology [9, 30, 3, 51, 49, 134]. Other examples of this architectural complexity have included navigational patterns, community structures, congestion, and other social phenomena resulting from users' behavior [79, 80, 4, 106, 107]. Prominent among these properties have been the in-degree and out-degree distributions of the Web graph, both of which turn out to have heavy tails. In the case of in-degree, the research cited shows close fits to power-law distributions over several orders of magnitude, results which are confirmed by the present work.

While these insights have informed the design of a variety of Web-related applications such as crawlers and caching proxy servers, structural analysis of the Web graph has seen its greatest application in ranking pages returned by search engines. In particular, the well-known PageRank [29] and HITS [90] algorithms use the pattern of links connecting pages to rank them without needing to process their contents; these algorithms have inspired a vast amount of research into ranking algorithms based on link structure. Researchers have also shown that the structural properties of the link graph extend to the host graph, which considers the connectivity of entire Web servers rather than individual pages [23].

However, the unweighted Web graph does not represent user behavior; it is an artifact that

exists whether or not users actually traverse its links. As a philosophical matter, it is reasonable to ask ourselves whether a hyperlink that no human has never clicked on truly exists—or, if it does, whether it exists as genuinely as one that ferries many thousands of surfers from one page to another on every hour of every day. The link structure of the Web can and does differ greatly from the set of paths that are actually navigated, and Internet researchers have been quick to recognize that structural analysis of the Web becomes far more useful when combined with behavioral data. Network managers and capacity planners are accustomed to this view of the Web graph as a weighted network; numerous tools exist for analyzing server logs, determining trends in the quantity of HTTP traffic on a network, and so forth. These tools offer high-level insight into aspects of user behavior such as what pages are most popular, what referring sites are most common, what percentage of traffic in a local network is devoted to Web traffic, and so forth.

As described in Section 2.3, a variety of behavioral data sources exist that allow researchers to identify these highly traveled paths. A number of early research efforts used browser logs to characterize user navigation patterns [36], time spent at pages, bookmark usage, page revisit frequencies, and overlap among user paths [44]. The most direct source of behavioral data for individual sites comes from the logs of the servers themselves, which have been used for applications such as personalization [114] and improving caching behavior [149]. More recent efforts involving server logs have met with notable success in describing typical user behavior [71]. Because search engines serve a central role in users' navigation, their log data is particularly useful in improving search results based on user behavior [7, 102]. Adar *et al.* [5] used traffic information from instrumented browsers to study the patterns of revisitation to pages. A related approach involving a select panel of users has been used to describe the exploratory behavior of Web surfers [19].

I have already described some of the disadvantages of these log-based data sources as compared to information taken directly from the network. The Internet itself can serve as a rich source of data

on Web behavior. Besides my own work described in Chapter 3, network flow records have been used to identify some statistical properties of Web user behavior and discriminate peer-to-peer traffic from genuine Web activity [58].

In the end, the most detailed source of Web traffic will always be the traffic itself—in other words, packets captured from a running network. My use of this approach is inspired in part by the work of Qiu *et al.* [130], who used captured HTTP packet traces to investigate a variety of statistical properties of users' browsing behavior, especially the extent to which they appear to rely on search engines in their navigation of the Web. This study directly inspired the work on session identification described in Chapter 4, which demonstrates that a simple timeout-based approach as used by this study and others yields an inadequate definition of Web sessions.

While simply measuring Web traffic is an interesting and worthy project, a greater goal is to develop behaviorally plausible models for that traffic. Any such model of user activity on the Web has two main contributions. On one hand, we obtain insights on how people interact with the Web: as more and more people spend more and more of their time online, their Web traces provide an increasingly informative window into human behavior. On the other hand, a model is also of immense practical value: we gain a basis for exploring ways in which we can improve users' browsing experiences or design sites to better accommodate their habits. Applications range from ranking search results [29] to guiding crawlers [40] to predicting advertising revenues.

Despite its simplicity, the PageRank random surfer model [124] has been a remarkably robust model for human Web browsing, contributing directly to the rise of Google as the dominant search engine on the Web. However, as will be described in Chapter 4, the large volume of click data gathered for this work reveals serious shortcomings of this model. Traffic patterns aggregated across users show that some key assumptions—uniform random walk and uniform random teleportation—are widely violated, making PageRank a poor predictor of traffic. One suggestion

for overcoming these shortcomings has been to substitute actual traffic data for ranking pages [101]. However, this may create a feedback cycle in which traffic grows super-linearly with popularity, leading to a situation, sometimes called "Googlearchy," in which a few popular sites dominate the Web and lesser known sites are difficult to discover [125, 65].

In any case, simply accepting traffic data as a given quantity of mysterious origin does nothing to further our understanding of user behavior; the deficiencies of the random-surfer model must be rectified by improving the model itself. More realistic models have been introduced in recent years to capture relevant features of real Web browsing behavior, such as the back button [103, 27]. There have also been attempts to model the interplay between user interests and page content in shaping browsing patterns. One such study examines the correlation between changes in page content and patterns of revisitation [6]. Huberman *et al.* proposed a model in which pages visited by a user have interest values described by a random walk, with navigation continuing as long as the current page has a value above some threshold [80]. This kind of model is closely related to algorithms proposed for improving topical crawlers [105, 110, 106].

Analysis of the available data suggests the need for an agent-based model of Web activity that carries state information and can account for both individual and aggregate traffic patterns observed in real-world data. The development of such an agent-based model is described in Chapter 4. In this effort in particular, I am greatly indebted to the work of my collaborators, which includes but is by no means limited to the citations given here [110, 73, 72].

Finally, the Web analysis I describe also relates to established results in anomaly detection and anonymization software for the Web. The Web Tap project, for example, attempted to distinguish anomalous traffic requests using metrics such as request regularity and interrequest delay time [26], quantities which I discuss in Chapter 4. The success of systems that aim to preserve the anonymity of Web users is known to be dependent on a variety of empirical properties of behavioral data [145],

some of which I address directly.

3

Network flow data

3.1 Data source

The logical starting point for an effort to understand the behavior of users on the Internet is to examine their traffic: as much of it as possible, involving as many applications as possible, from as numerous and diverse a set of users as possible—and all of this without compromising their privacy. As we have already discussed, a full survey of traffic is untenable; the population of the Internet has grown beyond one billion users, and they exchange petabytes of data on a daily basis. Moreover, inspecting each one of their packets would constitute as egregious a violation of their online privacy as one could imagine. This chapter therefore begins with a discussion of the compromises and tradeoffs, both technical and social, made to obtain a data set that fulfills these goals as well as possible.

The Internet2 network

As explained in Chapter 2, if we wish to study a large number of Internet users while minimizing sampling bias, we are strongly advised to seek data from a transit network. Such a network

generally operates on behalf of a wide variety of customers and institutions, providing a conduit for data without being a significant consumer or producer of traffic in its own right. Unfortunately, there are few opportunities for gaining access to such data; the world of "Tier-1" Internet Service Providers is intensely competitive, and even high-level summaries of traffic can reveal a significant amount of strategically useful information. By studying flow summaries from a competitor's network, a rival ISP can infer some of traffic policies implemented, locations of possible congestion or over-provisioning, the identities of large customers, and so forth. Though transit networks do represent an ideal source of traffic data, it is difficult to fault their commercial owners for being reluctant to grant researchers access to their infrastructure.

Fortunately, a viable alternative exists, in the form of the Internet2 network [1] Internet2 (formerly called "Abilene"). This network is a modern counterpart to the National Science Foundation's *NSFNet*, which served as the primary Internet backbone for American academic and research traffic until the commercialization of the Internet in 1995. Despite the implications of the name, Internet2 is very much a part of the current Internet and uses the same protocol stack. It is a high-bandwidth TCP/IP network that spans the United States and provides unmetered connectivity to hundreds of colleges, universities, government labs, and research laboratories. The backbone of the network, illustrated in Figure 3.1, consists of 10-Gbps fiberoptic links connecting high-performance routers located in major metropolitan areas across the United States.

The Internet2 network carries a wide variety of traffic. Its user base includes, besides myriad graduate students and academic faculty and staff, hundreds of thousands of undergraduate students who are often early adopters of new network applications. In addition, Internet2 also provides transit for data from dozens of international academic and research networks, making it a

[1] This network was formerly called Abilene and was operated by the Internet2 Project. This organization now uses "Internet2" to refer to both itself and the network it operates.

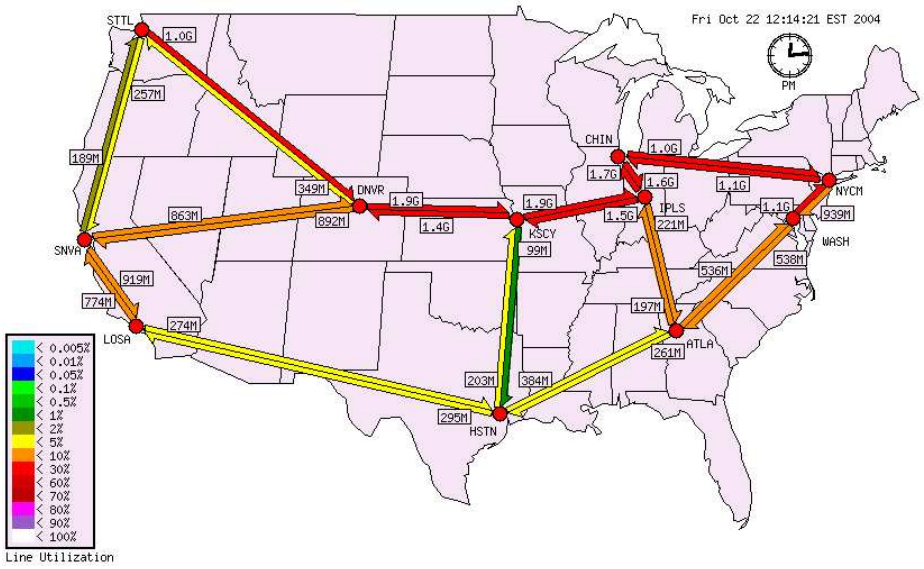
Figure 3.1: Map of the Internet2 network backbone, which connects hundreds of colleges, university, and other research institutions with high-speed fiberoptic links. The black dots indicate the core router nodes that generate network flow records. *(Map courtesy of Internet2 network.)*

major data path between Pacific Rim nations and Europe. This not only gives the traffic an international flavor, but also tends to reduce the intensity of the strong diurnal cycle often seen in regional networks. At the time the initial data sets described in this chapter were collected, the network nominally carried only academic and research traffic; participating institutions were required to maintain separate connections to the commodity Internet for exchange with commercial networks. The large population of residential students tends to work against this policy, as peer-to-peer file transfers between American university students are quite likely to traverse Internet2. This policy was later amended, and Internet2 now includes peering connections with commercial ISPs, so that one only end of each connection need involve an Abilene participant. As we shall see in section 3.3, this change did little to affect the basic properties of the traffic.

Besides carrying this diversity of traffic, the Internet2 network has always been over-provisioned. This is an important attribute: the fact that the backbone of the network is never congested, even during peak hours, offers us a chance to observe what users do when the network itself does not impede their behavior. Typical levels of IPv4 traffic in the Internet2 network contemporaneous to the time the initial data sets were collected can be seen in Figure 3.1A. A more recent view of Internet2 activity is shown in Figure 3.1B.

A core part of the mission of Internet2 has been to partner with researchers, not just to provide infrastructure, but also to enable the study of that infrastructure. Moreover, Indiana University enjoys a special relationship with the Internet2 Project, as it has operated the Internet2 Network Operations Center since the project's inception in 1999. My own experience in developing network management applications for the NOC allowed me to become quite familiar with the personnel and technologies involved in running the network. While I do not enjoy access to Internet2 data not available to other interested researchers, I am familiar with the procedures and policies involved in acquiring traffic information from Internet2.

A



B

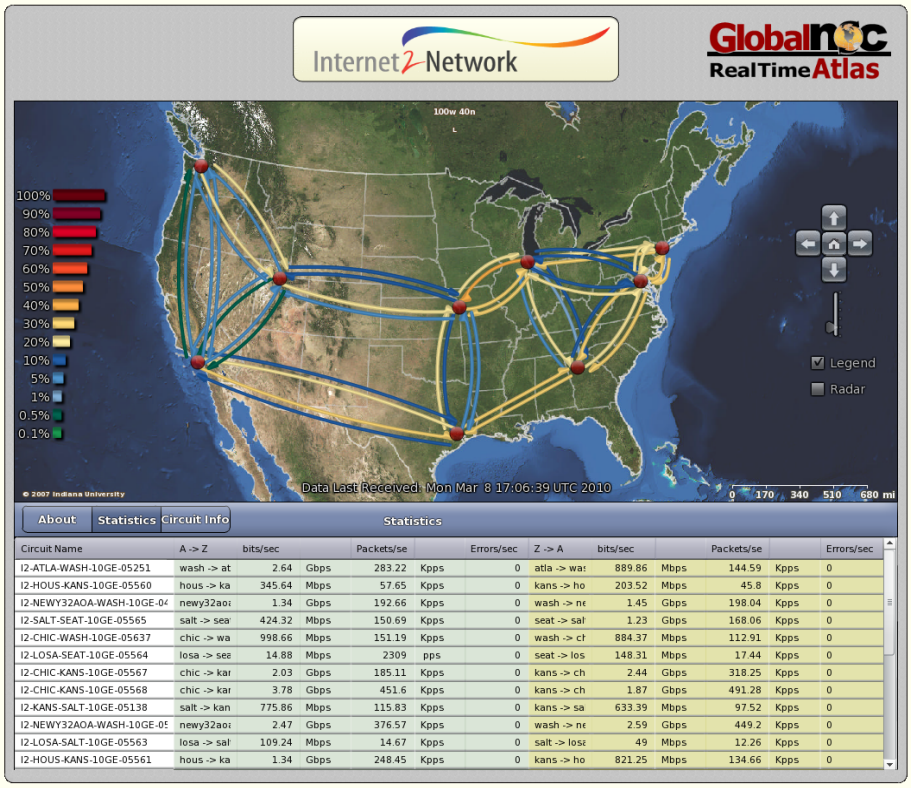


Figure 3.2: Typical levels of IPv4 traffic in the Internet2 network at the time of data collection for the preliminary study (A) and more recently (B).

For all of the reasons described, I believe that Internet2 is an appropriate choice for research of this type: it is an uncongested transit network, the user population is large and varied, and their organization is willing to partner with researchers as part of its core mission.

Netflow data

We have already discussed the intractability of deep packet inspection, especially for a modern network such as Internet2 that is made up of 10 Gbps links. Instead, Internet2 makes traffic information available in summary form through network flow data using Cisco's *netflow-v5* format, as mentioned in Chapter 2. Although Cisco Systems is the creator of this format, *netflow-v5* records have become a widely accepted standard and can be generated by routers from a variety of vendors. When I first began to gather flow records from Internet2, the core routers were from Cisco Systems. As part of a general network upgrade, these routers were replaced by models from Juniper Networks, Cisco's rival, without affecting the availability or format of network flow data.

Before we examine the structure of *netflow-v5* records themselves, we need to consider exactly how these flows are generated. It is not the case that each flow record generated by a router represents all the traffic generated by a single network conversation from its initiation to its termination. I have already mentioned a few of the reasons for this in Section 2.3.

First, flow records are unidirectional, whereas successful TCP connections always involve the passage of data in both directions. In the absence of packet sampling, every fully established TCP connection will generate two flow records.

Second, UDP and ICMP are connectionless protocols, which means that it is not necessarily the case that any two packets with the exact same endpoints have anything to do with each other, regardless of how close in time the router receives them. In practice, of course, this is nonsense:

a second packet from port 53 on host 10.0.0.1 to port 7331 on host 192.168.0.1 is almost certainly a continuation of the first one. Routers therefore apply the obvious heuristic by bundling UDP and ICMP packets with identical endpoints into the same flow, using a inactivity timeout to terminate this grouping. Absent conscious effort to generate spurious traffic, this heuristic is quite reliable. Indeed, most network address translation systems rely on this technique to handle UDP and ICMP traffic at all.

Next, even high-performance routers lack sufficient processing power to include every packet they process in their flow statistics. In order to keep the overhead associated with network telemetry to a reasonable level, routers generate flow data based on a sample of the data. In the case of Internet2, the core routers are configured with a sampling rate of 1 in 100 packets. The presence of sampling immediately raises concerns about bias, such as whether the sampling mechanism truly reflects a situation in which $p(sample) = 0.01$ for all packets, regardless of any attributes those packets may have. It also creates a situation in which not every flow is detected. There is a non-linear relationship between the number of packets in a flow and its chance of detection, which can raise concerns that any graph structure we derive from sampled flow data may not reflect the true characteristics of the underlying traffic. I explore these issues of bias in a quantitative way in Section 3.4; it suffices for now to state that the results presented there do support the methodology I describe in this chapter.

A final concern I have not mentioned until now also relates to router capacity. Memory has always been the scarcest and most valuable commodity in any router, leading network engineers to justly fear any feature that allows a router to accumulate state in an unbounded way. This means that routers are generally given a fixed allocation of memory for generating flow data, independent of current network conditions. In order to generate useful data about a continuously variable number of flows using an invariable memory allocation, routers periodically "flush" their current flow

statistics by generating flow records for ongoing flows whether they have been completed or not. Not only does this serve as the mechanism for implementing timeouts for UDP and ICMP flows, it means that long-lived TCP connections are likely to exported as flows several times during their existence. The number of flow records with common endpoints is thus not a meaningful statistic, which is one motivation for my hesitation to treat flows as rows of a relation.

Note that there is no coordination among the routers in any aspect of either sampling or flow generation. This means that a flow has an independent chance of generating another flow record for each router it traverses. In other words, a packet that touches the Los Angeles, Houston, and Atlanta core routers has a $p = 1 - (1 - 0.01)^3$ chance of generating at least one flow records, and may generate more. One can imagine some simple heuristics that remove "over-reported" flows by retaining flow records with identical endpoints from only one router over some period of time, but this would introduce a serious bias in favor of short flows that traverse multiple routers: because they are unlikely to be *detected* by more than one router, they are able to bypass the filtering heuristic entirely. Another approach is to maintain a configuration database for the Internet2 routers and use fields of the flow records to determine whether a flow is on its first or last Internet2 router, but this is a high-maintenance solution that requires additional access to the network infrastructure. I believe the safest solution, which I use in this work, is to simply accept this bias toward long-distance flows as a given and trust that there is no great difference in character between flows that traverse a single routers and those that traverse several. It is difficult to imagine how this assumption could be badly mistaken. We may overemphasize international flows between Asia and Europe, but this is a small share of the overall traffic carried by Internet2.

We are now ready to examine the exact mechanism through which routers deliver *netflow-v5* flow records to a management system. Flows are delivered using a simple connectionless protocol layered on top of UDP, using the packet format shown in Figure 3.1. In the interest of efficiency,

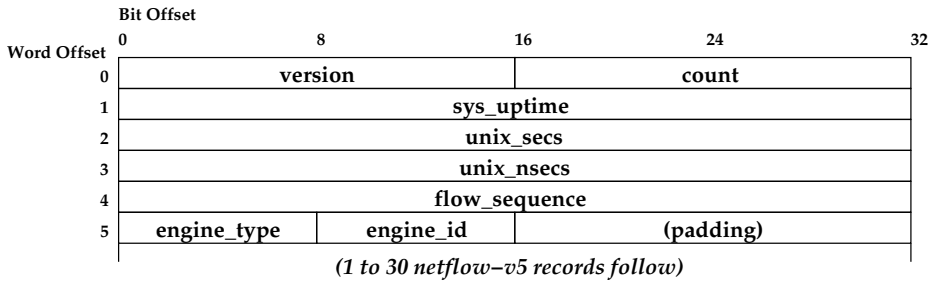


Figure 3.3: Format of a *netflow-v5* packet as encapsulated within a UDP datagram.

multiple flow records can be carried in a single packet, which consists of a header and as many as thirty flow records.

We now consider the individual fields of the header and their significance in analyzing flows.

- *version* (16 bits) is simply an indication of which export format is being used and is always 5 in the case of *netflow-v5*.
- *count* (16 bits) indicates how many flow records are present in the packet, which can be as little as one or as many as thirty.
- *sys_uptime* (32 bits) indicates the number of milliseconds since the router was booted. This information, together with the *unix_secs* and *unix_nsecs* fields, is necessary for associating absolute timestamps with a flow record.
- *unix_secs* (32 bits) is the standard "Unix epoch;" that is, it indicates the number of seconds since midnight on January 1, 1970 (UTC). This field and *unix_nsecs* form the only indicator of absolute time in a flow packet.
- *unix_nsecs* (32 bits) is a fractional count of nanoseconds. Most routers do not keep time to this degree of accuracy, so the lower-order bits are essentially meaningless.

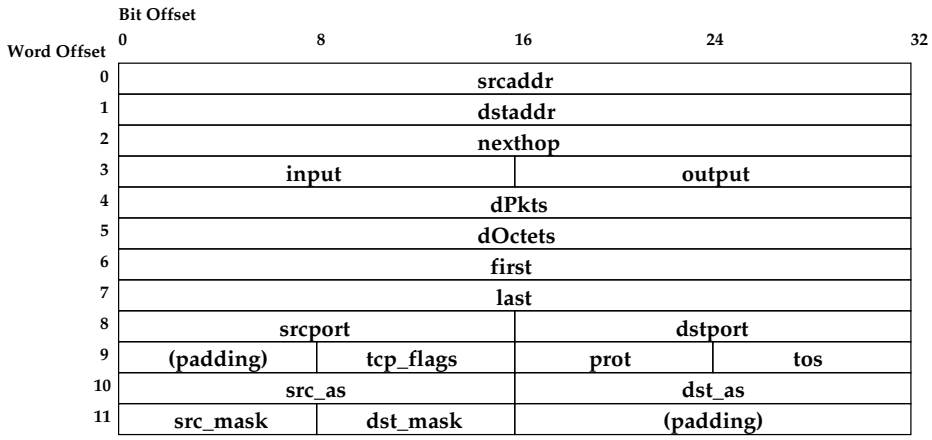


Figure 3.4: Format of an individual *netflow-v5* record.

- *flow_sequence* (32 bits) is a wrap-around sequence counter that is incremented by one for each flow record (not flow packet) exported from the router. Because *netflow-v5* is encapsulated by UDP, this is the only way for a flow collector to detect missing flow records.

- *engine_type* (8 bits) indicates the type of flow-switching engine that generated the records and is not relevant for our purposes.

- *engine_id* (8 bits) indicates which line card in the router generated the records and is again not relevant for our purposes.

The final two padding bytes in the header have been extended to include information about the sampling regime in effect on the router, but this information is not part of the original protocol definition and cannot be relied on to be present across multiple vendors' implementations.

Individual flows are represented by a fixed-length 48-byte record as shown in Figure 3.1. The meanings of the individual fields and their relevance to this analysis are described below.

- *srcaddr* (32 bits) is the IPv4 address of the system transmitting the data described by the flow.

- *dstaddr* (32 bits) is the IPv4 address of the system receiving the data described by the flow.
- *nexthop* (32 bits) is the IPv4 address of the next router to which the packets in this flow were delivered. In theory, this information can be used to detect when a flow is about to leave Internet2 or to detect asymmetric routing conditions, but I do not use it in this analysis.
- *input* (16 bits) is a Simple Network Management Protocol (SNMP) interface number that indicates on which port the packets in this flow entered the router. Interpreting this information properly requires SNMP access to the Internet2 routers, which is not generally available. In theory, it could be used to detect if this is the first Internet2 hop for the flow.
- *output* (16 bits) is an SNMP interface number that indicates on which port the packets in this flow exited the router. It is of comparable utility to the *nexthop* field.
- *dPkts* (32 bits) is an indicator of how many packets are associated with the flow. This is a rough measure of the burden the flow places on the routing infrastructure, but is a poor measure of the size of the flow, since the amount of payload in individual packets can vary greatly from application to application. Indeed, the ratio between packets and bytes transmitted may be useful in characterizing applications, though that idea has not been explored in this work.
- *dOctets* (32 bits) indicates the total number of bytes in the packets of the flow as measured at the network (IP) layer. This is the most appropriate level as which to meter flows, since the routing hardware does not work above the network layer and a single flow may traverse various link layers. This field is the primary measure of flow magnitude, and I use it to build up the weights of the edges in a behavioral network.
- *first* (32 bits) represents the number of milliseconds since the router was booted at the time the flow began. Changing this value into an absolute timestamp requires access to the *sys_uptime*, *unix_secs*, *unix_nsecs* fields from the flow header.

- *last* (32 bits) represents the number of milliseconds since the router was booted at the time the flow ended. As was the case for the *first* field, converting this information into an absolute timestamp requires access to the flow reader. Using the difference between *first* and *last* can give a biased estimate of the duration of the flow. The duration of a short flow will be underestimated because $last - first$ will be a measure of the time between the first and last *sampled* packets. The duration of long flows will be truncated based on the frequency at which routers flush their flow statistics.

- *srcport* (16 bits) gives the TCP or UDP source port number, or an equivalent demultiplexing key for other protocols (e.g., the ICMP message type). In general, this will be a well-known port number if the source host is a server and an ephemeral value if it is a client. However, recall that any application can be made to run on any port: most of the analysis in this chapter relies on the assumption that the majority of traffic is not masquerading in this way.

- *dstport* (16 bits) is the equivalent of the *srcport* field for the destination host. Again, this usually will be a well-known port number if the destination is a server and an ephemeral value if it is a client.

- *tcp_flags* (8 bits) is a cumulative logical OR of the TCP flags across all the packets in the flow. In a sampled environment, this offers a semi-reliable method of determining traits of some TCP flows. If the SYN flag is set, the flow includes the beginning of the connection; if the FIN flag is set, the flow includes the end of the connection; and if the RST flag is set, the connection was not fully established. In practice, this information is of marginal utility, since these flags are set in only one or two packets in any connection and are unlikely to be sampled.

- *prot* (8 bits) identifies the transport layer protocol specified in the IP header. As mentioned before, the vast majority of Internet traffic will use 0 (ICMP), 6 (TCP), or 17 (UDP). Recall

also that UDP and TCP have completely orthogonal port definitions, so this information is an integral part of the flow.

- *tos* (8 bits) duplicates the IP type of service (ToS) byte from the IP header. This byte has been defined in a variety of ways over the years and cannot be used as a reliable measure of traffic in a transit network.

- *src_as* (16-bit) gives the AS number of the source network. This information is generally available only from routers at the edge of an AS (i.e., those running Border Gateway Protocol). In the case of Internet2, all of the core routers are also edge routers, so this information is reliably present.

- *dst_as* (16-bit) gives the AS number of the destination network. Again, this information is available only at the edge of a network, but all Internet2 routers are at the edge of its network.

- *src_mask* (8-bit) gives the prefix size associated with the source network. This is the number of bits in the source IP address that identify the network part of the address and is the same quantity as is used for writing an IP prefix in Classless Inter-Domain Router (CIDR) notation. For example, if the source address is 156.56.103.1, which is part of the 156.56.103.0/24 network, this value would be 24. This number can be used to detect whether critical information has been lost through one of Internet2's allowable anonymization schemes, which are described in Section 3.1.

- *dst_mask* (8-bit) is the equivalent of the *src_mask* field for the destination network.

We can see from this discussion that some of the fields are either of little use in a sampled environment or are not useful for the analysis described here and may as well be discarded to save storage space. Also note that a flow record contains no absolute record of time; associating a timestamp with a flow requires use of the *sys_uptime* and Unix epoch information in the flow

header. These factors inform my decision to store flow records not in their original state, but in a derived format, as described in the next section.

Data collection

Flow records do not contain any information above the transport layer; they contain no details of application protocols and do not provide any payload contents. Nevertheless, because they include the IP addresses and transport protocol port numbers for both hosts, they do contain enough information to raise personal privacy concerns. Even though virtual hosting is widespread on the Web, a flow record can easily provide evidence that a particular workstation is browsing adult Web sites. Flows also provide evidence that hosts are participating in peer-to-peer networks, running potentially vulnerable services, or have been compromised by network intruders. Many of these situations represent actual crimes, which raises an age-old ethical question for researchers of human behavior: at what point is the researcher obligated to intervene rather than observe? Even if researchers have no mandate to act, the very fact the illegal activity can be inferred from flow records places responsibility on the custodian of the information, since they can be legitimately requested by a law enforcement organization in the course of a criminal investigation. All of these factors combine raw flow records sensitive enough a form of data so that additional security precautions must be taken on any system storing or forwarding the information.

Because of the desire to both protect the privacy of users and relieve researchers of the great responsibility implicit in storing raw flow records, Internet2 does not allow the research community to gather uncensored flow information. Under their "no spinning media" guideline, all flow records received by an analysis system must be anonymized before they are written to disk or other permanent storage.

The default method of anonymization is to zero out the lowest-order eleven bits of both the

source and destination IP addresses in each flow record. This reduces the 2^{32} (4 billion) possible IPv4 addresses to 2^{21} (2 million) different bins, each of which contains 2,048 possible addresses. This form of binning has the advantage of generally preserving organizational integrity: as long as a network is large enough to have a 21-bit prefix, all of its hosts are grouped together. While there are plenty of organizations smaller than this, it is certainly true of most universities and large companies. Unfortunately, with the introduction of commercial traffic to Internet2, many address blocks observed belong to ISPs with a great diversity of customers, none of whose traffic can be expected to bear any relation to one another.

Even if no such concerns existed, this level of grouping is far too coarse to study the behavior of individual users. This is especially true because we can expect a wide variation in the population density d of the 11-bit address blocks. As an example, MIT has around 10,000 students sharing 16 million IP address ($d \approx 0.0006$), while the Bloomington campus of Indiana University has around 40,000 students sharing 200,000 addresses ($d \approx 0.2$). Making any inference about the behavior of a typical individual would require detailed knowledge of this host density from institution to institution.

It is tempting to suggest the use of a one-way hash over IP addresses as an alternative means of anonymization, but the 32-bit address space of IPv4 is too small for such a scheme to provide any meaningful privacy. A modern workstation can easily perform a brute-force search through the entire key space to determine the true address of a host, or even compute and store a complete reverse mapping. Even if the address space were much larger, as in the case of IPv6, the ability of an attacker to inject spoofed packets following known patterns into the network makes it possible for them to confirm or disprove guesses as to a host's identity. For these reasons, Internet2 does not accept a long-term hashing algorithm as a means of anonymizing flow records.

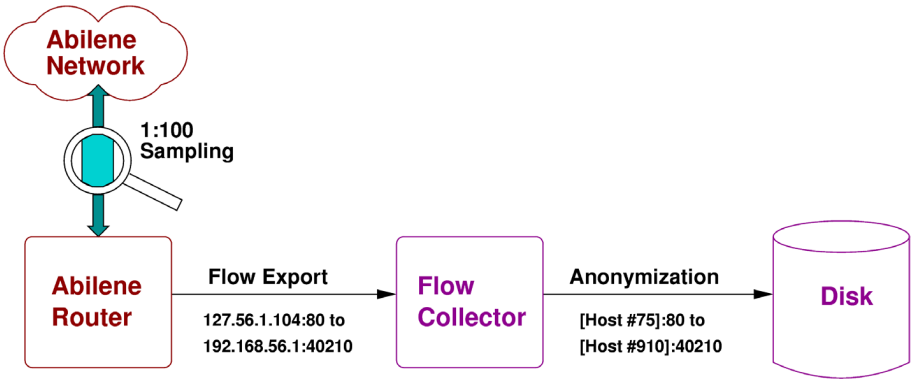


Figure 3.5: Overview of the data collection process for network flow data.

With these challenges in mind, I proposed a compromise solution, which was accepted by Internet2 and is used for all of the flow collection described in this work. Under the compromise scheme, the flow collection system maintains an in-memory index of IP addresses observed over the course of a 24-hour period: the first IP address observed is given index 1, the second is given index 2, and so forth. This mapping is consistent throughout the 24-hour period, but it is never written to disk and is discarded entirely at the end of the day. This disposal eliminates the possibility of searching the entire address space. It also greatly mitigates the power of injection attacks, which may be able to uncover the identities of hosts with extremely stable patterns of activity, but become unable to reverse-engineer transient activity in the absence of external corroborating information.

Figure 3.1 provides an overview of the data collection process. The raw flow records from the Internet2 core routers are sent to a flow collection system at the Advanced Network Management Laboratory at Indiana University, where they are reflected to my own data collection system. As flows arrive at this system, it anonymizes them using both schemes discussed above (clearing lower-order bits and maintaining a host index) and uses the information in the flow header to replace the *first* and *last* fields with absolute timestamps with one-millisecond precision. It also discards the padding bytes and rearranges the fields so that flow records are only 41 bytes long

rather than 48. These packed records are then written to disk for later analysis.

As might be expected, the number of flows generated by the Internet2 network on a daily basis increases over the long term as the amount of traffic increases, ranging from 700 million per day when I first began collecting data in 2004, to well over 1.3 billion today. When we consider the overhead of packet headers, one billion flow records corresponds to around 50 GB of data arriving at a mean rate of 4.6 Mbps. This volume of data is entirely within the ability of a well-connected workstation to gather, anonymize, and stream to disk without loss, and I have used the *flow_sequence* field from the flow headers to verify that the collection system does not miss flows with any regularity.

The final result of the collection process is a single file that contains all of the packed, anonymized flow records for a continuous 24-hour period. The analysis of behavioral networks described in this chapter is based on several representative sets of this data, all gathered from Internet2 on a typical weekday beginning at midnight Eastern Standard Time, and all containing one full days' worth of data. Data set A was gathered on September 30, 2004, and is the basis of the preliminary study. Data sets B and C were gathered on April 14, 2005, and April 22, 2008, respectively, and these are the basis of the extended study discussed in the remainder of the chapter.

Analytical tools

As discussed Chapter 2, the number of hosts represented in a single day's worth of Internet2 flow data turns out to be beyond the capability of most existing graph analysis software to work with, especially when this research began. Existing flow analysis packages were also inapplicable because they lacked either the capabilities to work with so many flows or sufficient flexibility to support the study of flow data from a graph-oriented perspective. The analysis of Internet2 flow data therefore necessitated the development of a variety of custom tools for gathering, processing, and analyzing flow data. These tools are described in detail in Appendix A; they are mentioned

here in passing as part of the analytical process, without discussion of their internal structure.

The primary challenge is to take a series of flow records as input and produce a behavioral network as output. Recall from Section 2.4 that a behavioral network is a weighted bipartite digraph that represents network flows aggregated over a period of time. Each node is either a client or server, depending on whether it initiated the connection, and the weight of an edge reflects the volume of data transfer between the hosts. This bipartite labeling is vital, since many popular applications, the Web among them, have highly asymmetric design in which clients and servers play enormously different roles. This labeling is also absent from flow records in *netflow-v5* format, which makes no indication as to which host is the client and which is the server in any particular flow.

A key concern is thus to recover the client and server labels from unlabeled flow records. The *tcp_flags* field is potentially of use for this purpose, since the initial packet from a client will be marked with only a SYN flag (as opposed to the server's response, which will be marked with both SYN and ACK). However, this idea is not worth pursuing: *tcp_flags* works in a cumulative fashion, so it is of use only for short flows in which no acknowledgments from the client to the server have been detected. The initial SYN packet is also unlikely to be sampled, and the approach is of no utility whatsoever for UDP traffic.

We can instead make use of the fact that TCP stacks are instrumented so that the client uses an ephemeral port number, whereas the server's port number must be known in advance. If we observe a flow with endpoints at TCP ports 16974 and 80, we can reasonably conclude that the latter port represents the server, since 80 is the well-known port assignment for HTTP. We do not need to consult the list of Internet Assigned Numbers Authority port assignments to apply this principle. In fact, such an effort would be counter-productive: there are many network applications with formally assigned port numbers that are never used to any appreciable extent, there are applications

with informally assigned numbers that are wildly popular, and we must decide what to do with flows in which both port numbers appear in the list. Moreover, many of the port numbers used on an informal basis by popular applications are also in the range that various operating systems use as ephemeral port numbers.

In resolving this situation, I invoke the axiom that when studying user traffic, observed behavior offers a more accurate view of network conditions than any standards document or protocol specification. To this end, I have a preprocessing stage in which I generate a tally of the number of flows that reference each TCP and UDP port. Having this tally allows me to apply a simple heuristic to identify the client and server in each flow: the server is the one with the more commonly used port number. This approach is imperfect, since it can incorrectly classify traffic with spoofed client ports or genuine uses of ephemeral port numbers that overlap with customary usage. However, it has the great advantage of believing the evidence from the network itself as to what the major network applications are on any given day. If a major peer-to-peer network moves to another port number, we capture that movement immediately without having to become aware of it through other channels. The smaller the amount of potentially obsolete domain knowledge we manually apply to observed behavior, the better.

Note that it is quite possible for a single Internet host to behave as both a client and a server, even in the context of a single application. For example, a user who both maintains a personal Web site and surfs the Web on their workstation fits in this category. The degree of overlap varies from application to application, but in no case can we be assured of a complete separation between clients and servers. Furthermore, the labeling heuristic means that we cannot distinguish clients from servers until after we have built the host index, so both clients and servers are labeled in the same namespace. The weighted adjacency matrix for a behavioral network is therefore stored in two files: one that holds weighted edges directed from servers to clients, which I refer to as *STOC*;

and one that holds weighted edges directed from clients to servers, which I refer to as *CTOS*.

The tool just described handles only the process of deriving STOC and CTOS files from a set of flows. Other programs are responsible for aspects of analysis such as partitioning flows based on TCP ports so that we can examine the behavioral networks of different applications, extracting distributions from the STOC and CTOS files, and supporting the analysis of those distributions. These tools operate in a strictly defined way without the need for heuristics of the type described above and do not require more examination here.

One further application, *flowseek*, is not part of the normal analytical work flow, but its interactive use has helped to guide much of my flow-related research. This application, also described in Appendix A, implements a query language for flexible and high-speed extraction of matching records from repositories of flow data. The functionality of *flowseek* is similar to that of existing packages such as the already mentioned *flow-tools*, but its query optimization and clear syntax have made it far more suitable for interactive use and data exploration.

3.2 Preliminary study

This preliminary study uses data set A and examines only the static properties of the behavioral network of the Web, without reference to any temporal properties, other applications, or Internet traffic as a whole. Many of its findings were unexpected and novel, and they were a direct inspiration for the more detailed study described next.

As mentioned before, data set A was gathered on September 30, 2004, beginning at midnight EST. During the twenty-four hours of data collection, the analysis system received and saved to disk approximately 742 million flow records involving almost 30 million individual hosts. Of these flows, about 319 million (43%) involved Web traffic, which was defined as any connection with

Table 3.1: Dimensions of the Web behavioral network in the preliminary network flow study.

Subset	Clients	Servers	Both Roles
CTOS	17,700,000	363,000	29,000
STOC	2,380,000	148,000	17,500

an endpoint at TCP port 80. This definition excludes HTTPS (port 443) traffic and activity on the alternate HTTP port (port 8080). It also includes some small proportion of traffic not related to the Web but nevertheless using port 80 to evade policy or security constraints. The Web behavioral network induced by even this restricted set of flows is enormous, as shown in Table 3.1.

Keeping the graph segregated into the CTOS and STOC components allows us to study the degree k and strength s of clients and servers independently. The CTOS component describes the out-degree $k_{out,C}$ and out-strength $s_{out,C}$ of clients and the in-degree $k_{in,S}$ and in-strength $s_{in,S}$ of servers. Similarly, the STOC component describes the out-degree $k_{out,S}$ and out-strength $s_{out,S}$ of servers and the in-degree $k_{in,C}$ and in-strength $s_{in,C}$ of clients. We first consider the properties of clients and then examine servers.

Web clients

In examining client traffic in the Web behavioral network, we are concerned primarily with properties that have implications for the design and modeling of browsers, crawlers, and other user agents.

One such property is clearly the number of servers $n_S(C)$ with which a client C interacts, which is the same as its undirected degree k_C. This is by no means a normal distribution: the mean is $\langle n_S \rangle = 3.52$ servers per client, but the standard deviation $\sigma(n_S) = 35.1$ is a full order of magnitude larger. The heavy-tailed nature of the distribution n_S is made clear by Figure 3.6, which shows an

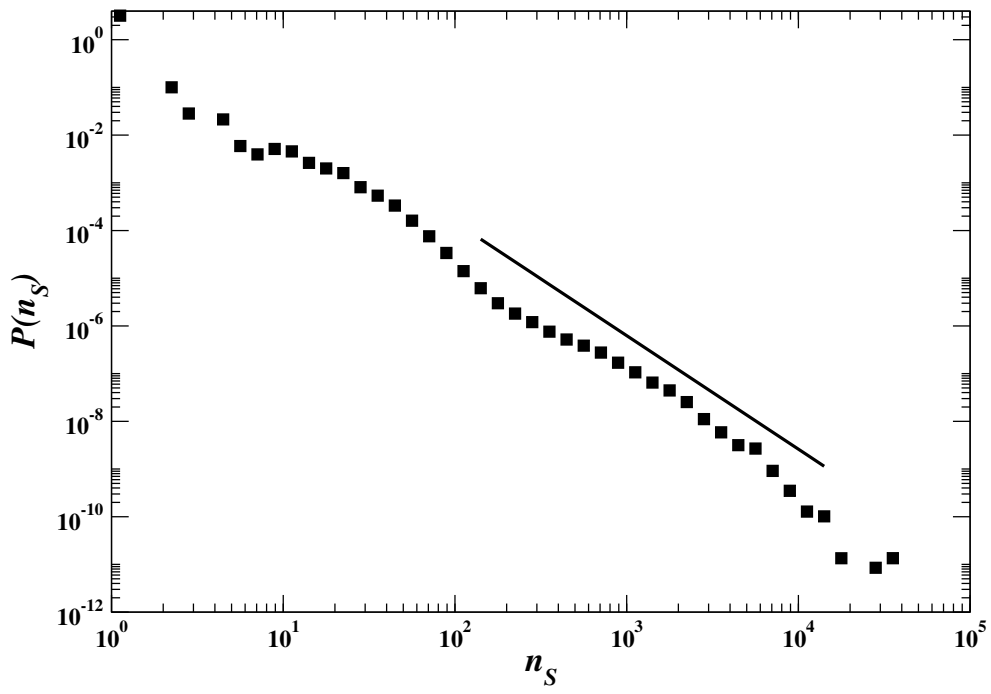


Figure 3.6: Probability distribution of the number of servers n_S contacted by a Web client. The solid line indicates the power-law behavior with exponent -2.4.

estimated PDF. The data are well-approximated by a power law $P(n_S) \sim {n_S}^{-\gamma}$ over several orders of magnitude. When $\gamma < 3$, as in this case, the variance of the distribution diverges, implying that the standard deviation is not an intrinsic value of the distribution and is bounded only by the size of this sample. The mean value $\langle n_S \rangle$ is therefore of little practical value. We lack a characteristic number of servers per clients and have an appreciable probability of finding clients that contact an enormous number of servers, with no clear cutoff in the distribution.

As a confirmation of this scale-free behavior of client connections, we can examine the distributions $k_{in,C}$ and $k_{out,C}$ independently. The former describes the number of servers that contacted the client, and the latter describes the number of servers contacted by the client. We would expect to find that one or both of these distributions would be scale-free as well, and we indeed find that $\gamma_{in} \simeq \gamma_{out} \simeq 2.4$, consistent with the above result.

A second important measure of client behavior is the total amount of data sent $s_{out,C}$ and received $s_{in,C}$ when interacting with servers, since the performance of a browser is directly related to its typical workload. In this case, however, we again find that the distributions are extremely broad and span nine orders of magnitude, as shown in Figure 3.7. Once again, the tails are well-approximated by power laws over four orders of magnitude, where $P(s_{out,C}) \sim s_{out,C}^{-\alpha_{out}}$, with $\alpha_{out} = 2.1 \pm 0.1$ and $P(s_{in,C}) \sim s_{in,C}^{-\alpha_{in}}$, with $\alpha_{in} = 2.2 \pm 0.1$. While it is worth keeping in mind that sampling means that the actual traffic values are roughly two orders of magnitude larger, this multiplicative factor affects neither the shape of the distribution nor the power-law fit. In the case of in-strength (data received by clients), we find a mean of $\langle s_{in,C} \rangle = 9.28 \times 10^4$ and standard deviation of $\sigma(s_{in,C}) = 2.05 \times 10^6$. In the case of out-strength (data generated by clients), we find a mean of $\langle s_{out,C} \rangle = 1.11 \times 10^3$ and standard deviation of $\sigma(s_{out,C}) = 1.44 \times 10^5$. In both cases, the standard deviation is two orders of magnitude larger than the mean, which underscores the lack of any characteristic value and the extreme heterogeneity in the amount of data handled by Web clients. Especially striking is the evidence for similar behavior for both incoming and outgoing traffic: if the Web were truly a broadcast medium, we would expect much greater asymmetry between the distributions.

The distribution of weights w_{CS} between clients and servers in the Web behavioral network also provides insight into client behavior. Each weight represents the aggregate volume of traffic between a specific client-server pair over the course of the day, allowing us consider the probability $P(w_{CS})$ that any given connection carries traffic w_{CS}. Figure 3.8 shows that the weights in the network also exhibit a heavy-tailed distribution that is well fit by a power-law distribution $P(w_{CS}) \sim w_{CS}^{-\delta}$ with $\delta \approx 2.3 \pm 0.1$. The large variability in this distribution provides evidence that extreme heterogeneity in traffic occurs even at the level of individual connections.

In order to get more insight into these patterns, we can examine the correlation between the

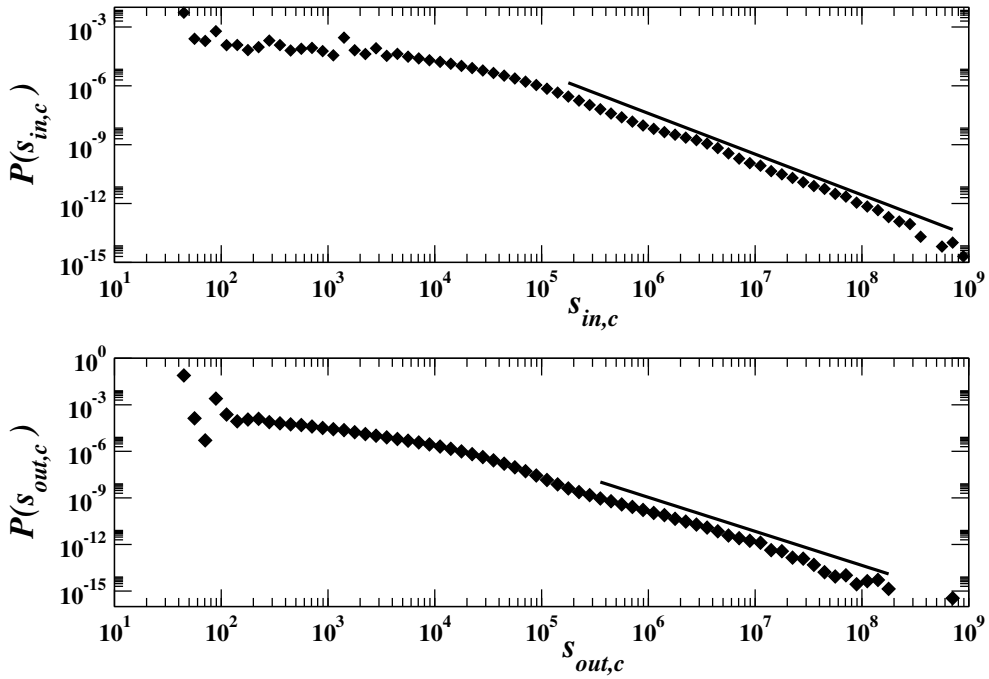


Figure 3.7: Probability distribution of the total incoming data $s_{in,C}$ and outgoing data $s_{out,C}$ of clients. The solid lines indicate power-law behavior with slopes of -2.2 and -2.1 for the upper and lower graphs, respectively.

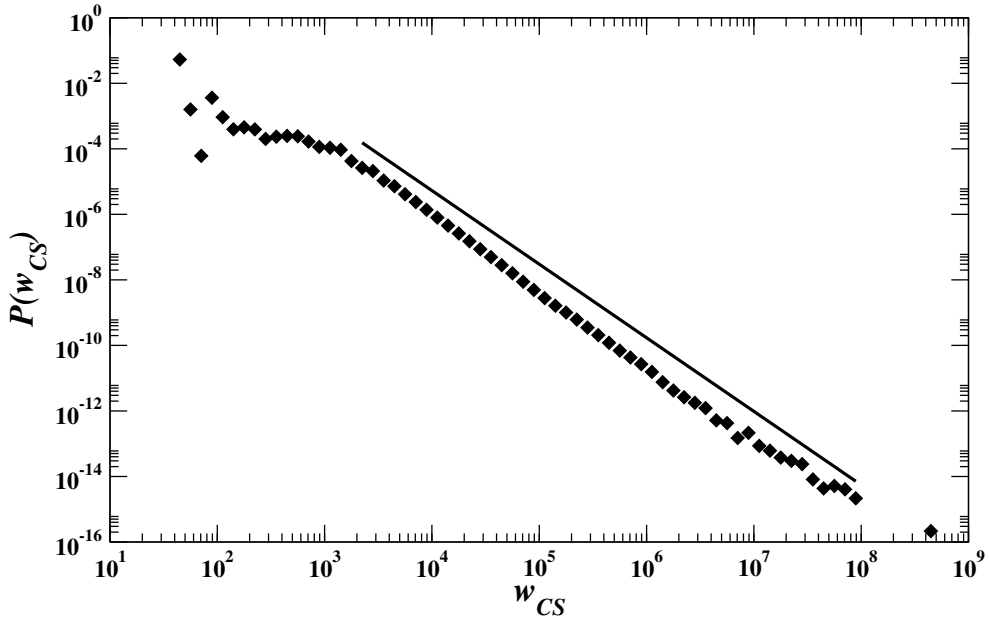


Figure 3.8: Probability distribution of aggregate client-to-server traffic for each (client, server) pair. The solid line has slope -2.4.

number of connections handled by clients and the total volume of those connections. We intuitively expect that in-strength $s_{in,C}$ ought to behave as an increasing function of in-degree $k_{in,C}$, and similarly that out-strength $s_{out,C}$ behaves as an increasing function of out-degree $k_{out,C}$, simply because every connection must involve the transmission of some volume of data in order to exist at all. From this perspective, the power-law character of the distributions $s_{in,S}$ and $s_{out,S}$ may seem less surprising, since they can arise directly from the power-law behavior of the degree distributions. However, we find a more complex story when we examine the relationship between degree and strength. Figure 3.9 illustrates the behavior of the mean in-strength $\langle s_{in,C}(k_{in,C})\rangle$ for clients with in-degree $k_{in,C}$, and Figure 3.10 shows the analogous relationship for out-strength and out-degree. In both cases, we find that the increase in strength as a function of degree is yet again

well-approximated by a power law, yielding the relations

$$\langle s_{in,C}(k_{in,C})\rangle \sim k_{in,C}^{-\beta_{in}}$$

and

$$\langle s_{out,C}(k_{out,C})\rangle \sim k_{out,C}^{-\beta_{out}}$$

.

Performing a least-squares linear fit of the curves in the upper plots gives us the estimates $\beta_{in} = 1.2 \pm 0.1$ and $\beta_{out} = 1.2 \pm 0.1$. These exponents are a signature of non-trivial correlation between the number of connections handled by a client and its aggregate volume of traffic, a behavior that is both novel and unexpected. If $\beta < 1$, we have a situation in which the amount of data exchanged with each Web server declines as a client contacts more and more servers; this is akin to library patrons who check out many books being unable to finish all of them. If $\beta = 1$, there is no correlation between the number of servers a client communicates with and how much is said to each one, which we might expect. However, when $\beta > 1$ and we have superlinear scaling between degree and strength, as turns out to be the case, we have a situation in which the amount of data exchanged with *each* server increases as the number of contacts increases. This has clear implications for the design of scalable client applications: the total traffic load can be expected to grow exponentially as clients establish more connections.

The exponents β_{in} and β_{out} also allow a simple means of double-checking the measurements for the exponents of the degree and strength distribution by means of a scaling argument. Let us briefly consider all variables to be continuous rather than discrete, which is not unreasonable, since the distributions are many order of magnitude in width. If we substitute the scaling behavior $s(k) \sim k^{\beta}$ into the strength distribution $P(s)ds \sim s^{-\alpha}$, we find that $P(k)dk \sim k^{-\beta(\alpha-1)-1}dk$.

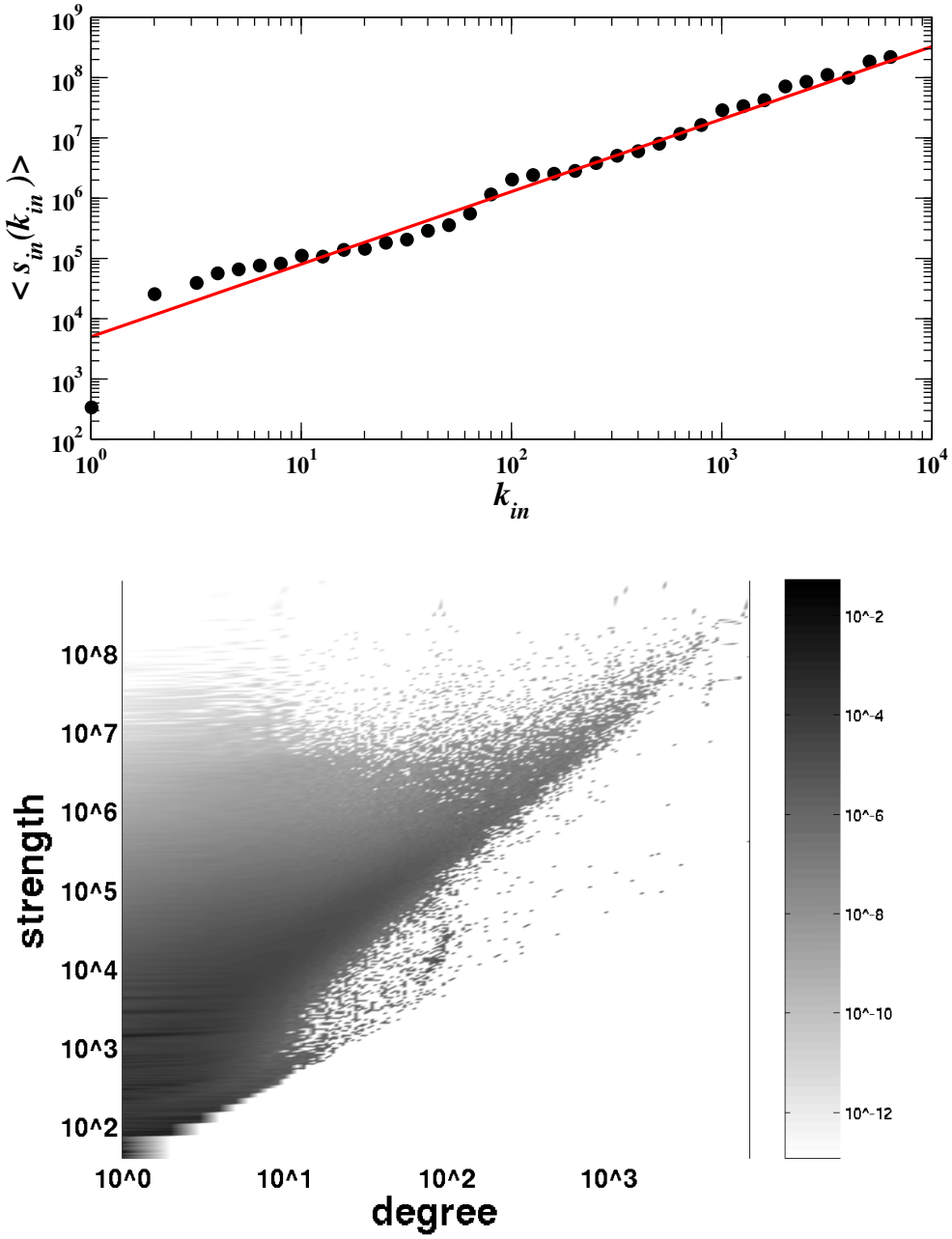


Figure 3.9: Behavior of incoming traffic to clients $s_{in,C}$ as a function of the number of server-to-client connections $k_{in,C}$. The upper plot shows the behavior of the mean in-strength $\langle s_{in,C}(k_{in,C})\rangle$ as a function of in-degree. The behavior is linear on a double logarithmic scale and in well-approximated by a power-law fit with slope $\beta_{in} \simeq 1.2$. The lower plot is a density map reporting the frequency of clients with a given in-strength for each value of in-degree. As discussed in Chapter 2, the tones in such a map represent frequencies of strength values, normalized within each degree bin, on a logarithmic scale.

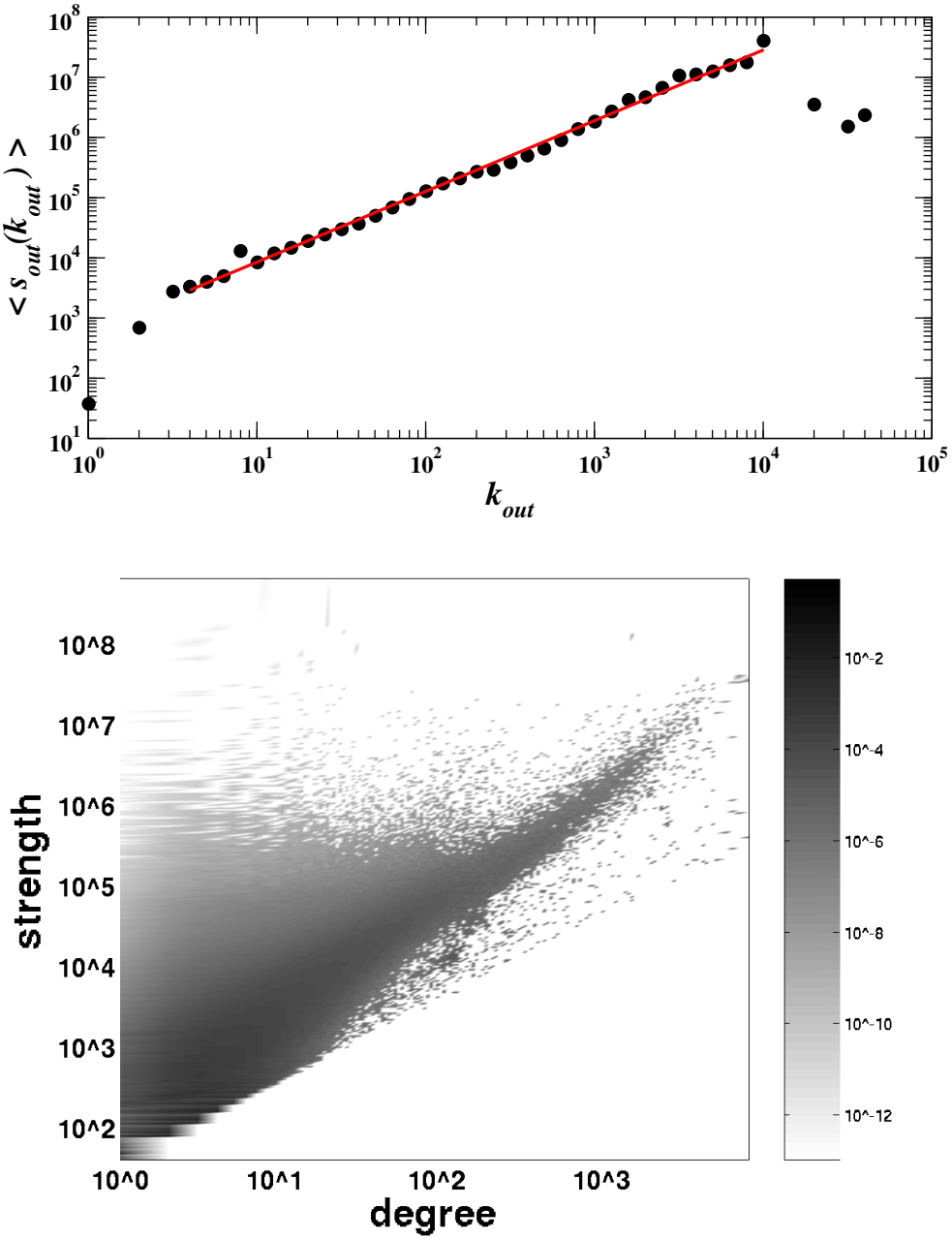


Figure 3.10: Behavior of outgoing traffic from clients $s_{out,C}$ as a function of the number of client-to-server connections $k_{out,C}$. As in the previous figure, the upper plot shows the behavior of the mean strength as a function of degree, and the lower plot is a density map of the degree-strength relationship.

However, by definition, we have $P(k)dk \sim k^{-\gamma}dk$. Equating the exponents in these expressions yields two equations relating α, β, and γ:

$$\beta_{out} = \frac{\gamma_{out} - 1}{\alpha_{out} - 1} \quad \text{and} \quad \beta_{in} = \frac{\gamma_{in} - 1}{\alpha_{in} - 1} \tag{3.1}$$

The values obtained empirically for the exponents satisfy these scaling relations within the margin of error for the estimates, which supports the internal consistency of these measurements.

Web servers

Having examined the role of clients in the Web behavioral network, we now consider servers. While this is less important in terms of understanding the behavior of individual users, information on the typical demands placed on a Web server is instrumental to both server and network design, as well as modeling Web traffic as a whole. The analysis in this case follows much the same direction as for clients.

We begin in analogous fashion by considering the number of clients $n_C(S)$ that each server S handles, which is equivalent to the undirected degree k_S. There is again strong evidence of an extremely heavy-tailed distribution: the mean is 142 clients per server, and the standard deviation is 2.34×10^4, over two orders of magnitude larger. Figure 3.11 illustrates the probability distribution $P(n_C)$ describing the likelihood that a server handles n_C clients. Given our previous experience and the large deviation, it should come as no surprise that the distribution is well-approximated by a power law. What is more surprising is the exponent we measure: we find that $P(n_C) \sim n_C^{-\gamma}$ with $\gamma \approx 1.8 \pm 0.1$. We considered earlier the case in which $\gamma < 3$, which indicated unbounded variance; when $\gamma < 2$, as is the case here, both the mean and the variance diverge and are limited only by the size of the sample and the finite size of the Web itself. In this regime, the mean number

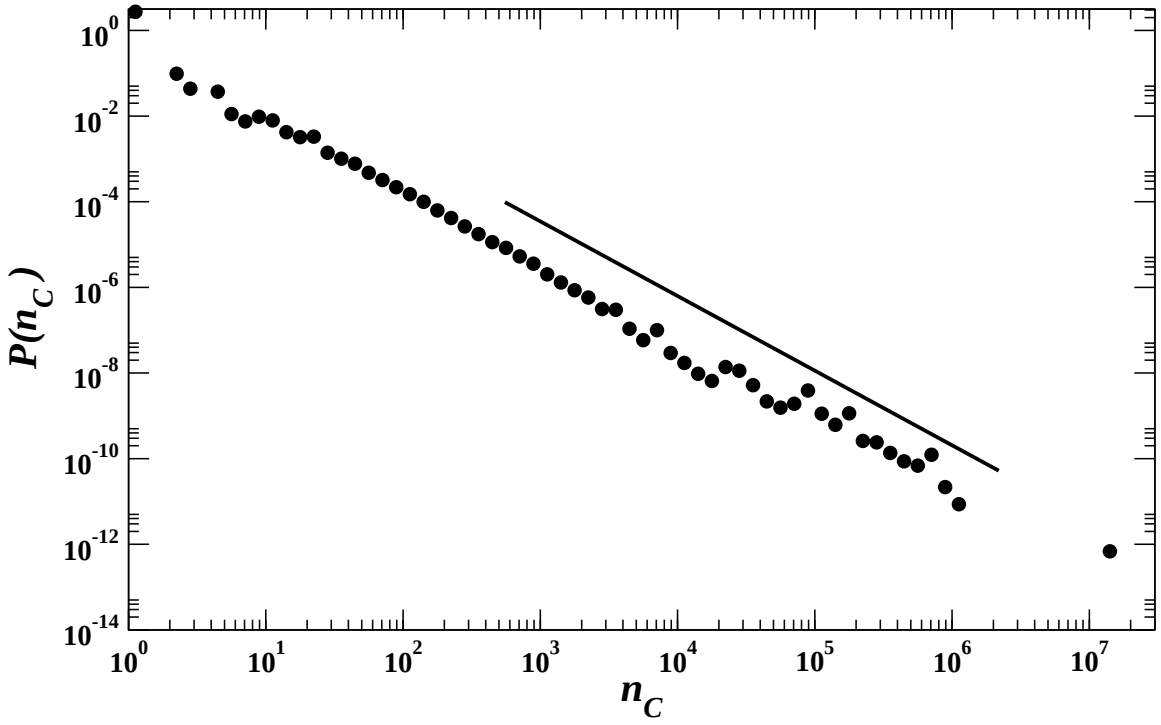


Figure 3.11: Distribution of the number of clients n_C handled by a Web server. The solid line indicates the power-law behavior with exponent -1.8.

of connections is no longer an intrinsic quantity of the system. This extreme heterogeneity is not usually found in constructed networks. It provides no evidence of a central tendency in server connections and no hint as to the scale to which it is most appropriate to target the design of a general-purpose Web server.

As in the case of clients, we can break $n_C(S)$ into its directed components $k_{in,S}$ and $k_{out,S}$ to see if the extreme heterogeneity is explained primarily by incoming or outgoing connections. We observe that both $P(k_{in,S})$ and $P_(kout, S)$ are described by power-law behavior with exponents $\gamma_{in} \simeq \gamma_{out} \simeq 1.8$, which is consistent with undirected degree.

We now turn to the server-based strength distributions. In this case, in-strength represents the amount of data that each Web server has received from its clients: this is the aggregate volume

of requests, form postings, and so forth. Once again, sampling results in underestimation of the true volume of the traffic but preserves the shape of the distribution. We find a mean value of 8.42×10^4 and a standard deviation of 5.41×10^6, once again suggesting a very wide distribution. The out-strength of servers represents the amount of data sent to Web clients by each server, which correspond to the aggregate size of Web pages, style sheets, images, and so forth. This time we find a mean of 1.35×10^6 and standard deviation of 3.91×10^7. These distributions, shown in Figure 3.2, are once again well-approximated by power laws $P(s_{in,S}) \sim s_{in,S}^{-\alpha_{in}}$ and $P(s_{out,S}) \sim s_{out,S}^{-\alpha_{out}}$ with $\alpha_{in} = 1.7 \pm 0.1$ and $\alpha_{out} = 1.8 \pm 0.1$. Since these exponents are definitely below two, we again face a situation in which we lack any central tendency. Because both the first and second moments of the mean diverge, there is no typical amount of either incoming or outgoing data that we can expect any random Web server to handle over the course of a day. From the standpoint of servers, Web traffic has no characteristic scale at all—a key finding when it comes to any attempt to model Web traffic or define acceptable levels of use.

The distribution of weights w_{SC} from servers to clients represents the amount of data sent from a particular server to a particular client over the course of the day. Given the findings just presented, it likely comes as no surprise that the probability distribution $P(w_{SC})$, depicted in Figure 3.2, is yet again well-approximated by a power law distribution. In this case, we find $P(w_{SC}) \sim w_{SC}^{-\delta}$ with $\delta = 2.3 \pm 0.1$. Once again, the dramatic heterogeneity in traffic manifests even at the level of individual host pairings.

As in the case of clients, we can gain some insight into the nature of the Web behavioral network by examining the interplay between degree and strength for servers. We can simply apply the same methodology to server nodes and examine the total amount of data received from clients $s_{in,S}$ as a function of the number of clients handled $k_{in,S}$, as shown in Figure 3.14. The behavior again

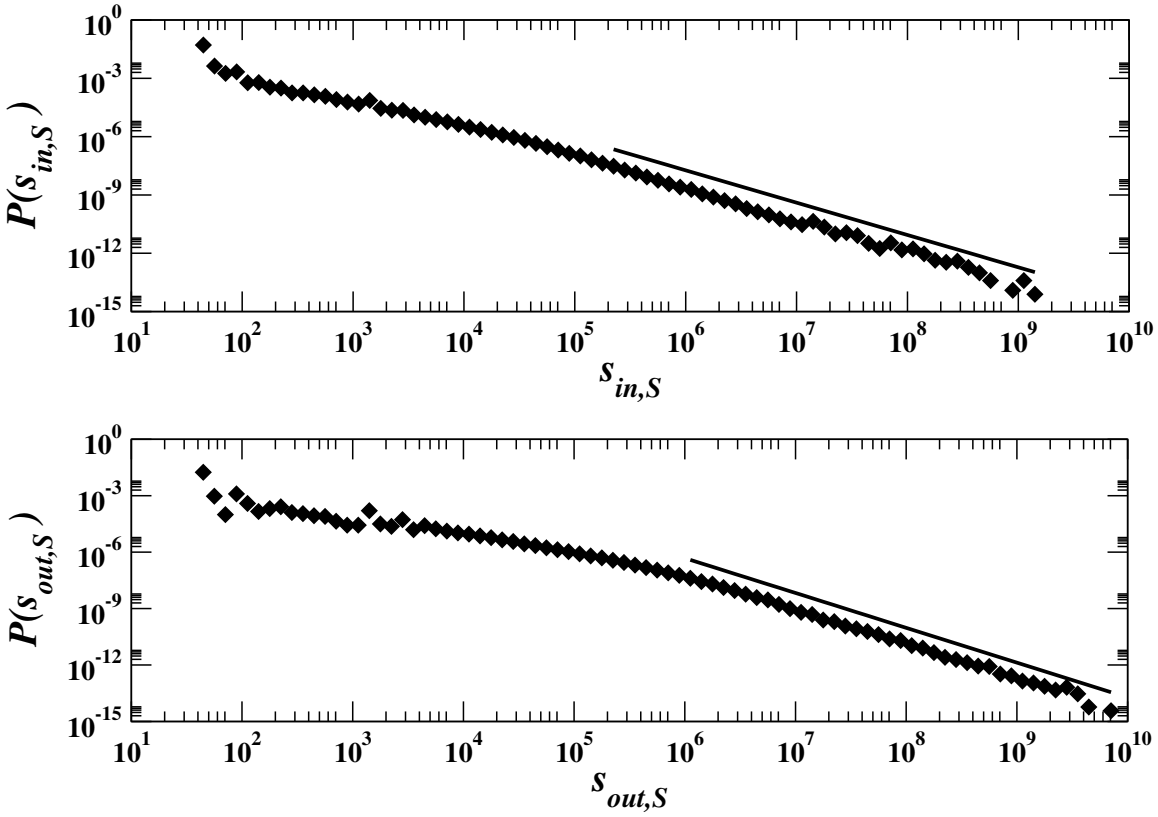


Figure 3.12: Probability distribution of the total incoming data (in-strength, $s_{in,S}$) and outgoing data (out-strength, $s_{out,S}$) of Web servers. The solid lines indicate power-law behavior with slopes of -1.7 and -1.8 for the upper and lower graphs, respectively.

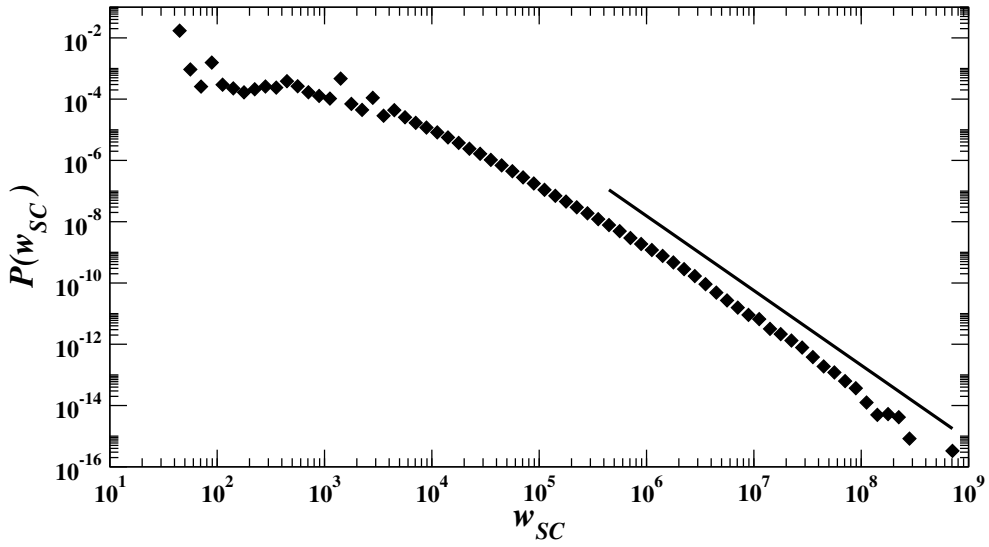


Figure 3.13: Probability distribution of aggregate server-to-client traffic for each (client, server) pair. The solid line has slope -2.3.

appears to be roughly linear on a double-logarithmic scale and can be modeled by the relation

$$\langle s_{in,S}(k_{in,S}) \rangle \sim k_{in,S}^{-\beta_{in}}$$

with $\beta_{in} = 0.9 \pm 0.1$. Similarly the relationship between the total amount of data sent to clients $s_{out,S}$ and the number of clients handled $k_{out,S}$ is shown in Figure 3.15. This relationship can be modeled in analogous fashion with exponent $\beta_{out} = 0.9 \pm 0.1$. These estimates reveal a qualitatively different behavior for servers as opposed to clients: instead of a super-linear relationship ($\beta > 1$), the data support a linear or sub-linear relationship ($\beta \leq 1$). This suggests a more conventional coupling between traffic and the number of connections in which the data per client does not increase more rapidly than the number of clients.

Just as was the case for clients, the scaling exponents β_{in} and β_{out} for incoming and outgoing connections of servers are predictable from the exponents α and γ of the strength and degree

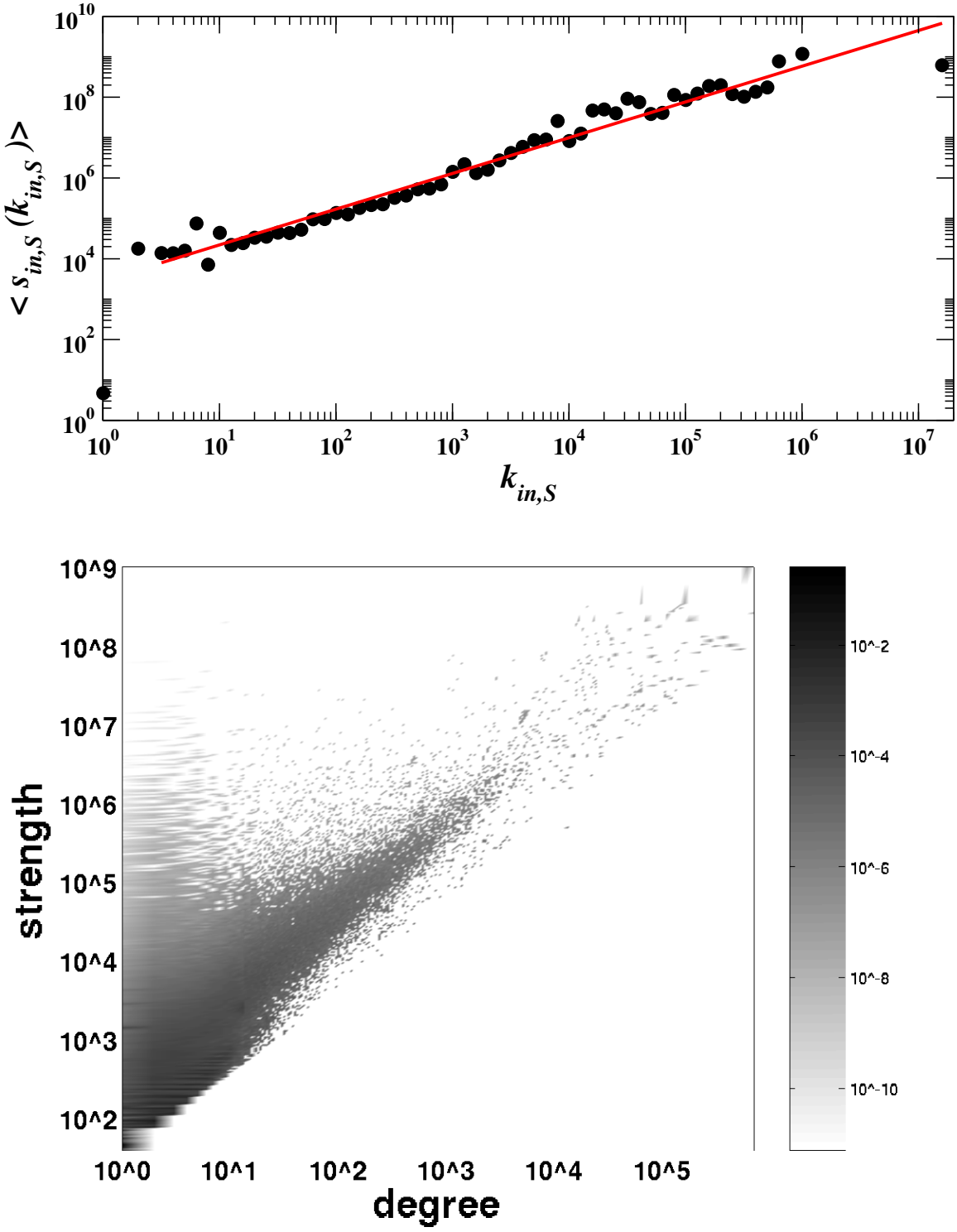


Figure 3.14: Behavior of incoming traffic to servers $s_{in,S}$ as a function of the number of client-to-server connections $k_{in,S}$. The upper plot shows the behavior of the mean in-strength $\langle s_{in,S}(k_{in,S})\rangle$ as a function of in-degree. The behavior is linear on a double logarithmic scale and can be approximated by a power-law fit with slope $\beta_{in} \simeq 0.9$. As before, the lower plot is an equivalent density map reporting the frequency of servers with a given in-strength for each value of in-degree.

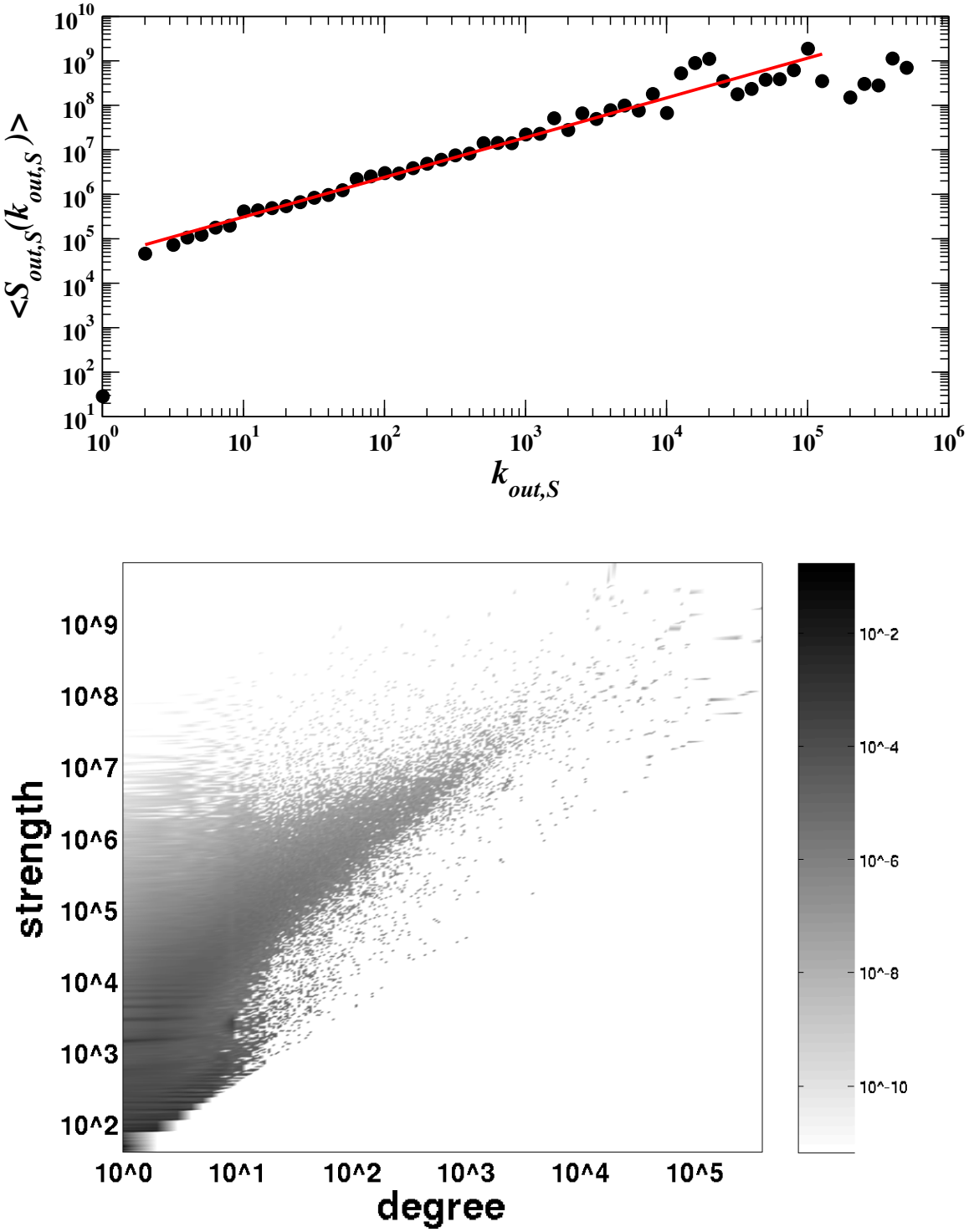


Figure 3.15: Behavior of outgoing traffic from servers $s_{out,S}$ as a function of the number of server-to-client connections $k_{out,S}$. Once again, the upper plot illustrates the mean strength as a function of degree, and the lower plot is a corresponding density map of the degree-strength relationship.

Table 3.2: For each variable x, we report the mean $\langle x \rangle$ and standard deviation σ. In all cases, we observe that the standard deviation exceeds the mean by one or two orders of magnitude. Also shown is the best-fit exponent of the power law approximation for the tail of the distribution.

Variable x	$\langle x \rangle$	σ	**Exponent**
n_S	3.52×10^0	3.51×10^1	2.4 ± 0.2
$s_{in,C}$	9.28×10^4	2.05×10^6	2.1 ± 0.1
$s_{out,C}$	1.11×10^3	1.44×10^5	2.2 ± 0.1
n_C	1.42×10^2	2.34×10^4	1.8 ± 0.1
$s_{in,S}$	8.42×10^4	5.41×10^6	1.7 ± 0.1
$s_{out,S}$	1.35×10^6	3.91×10^7	1.8 ± 0.1

distributions. The scaling relation presented earlier is again consonant with the fits.

Summary of results

The primary contribution of this initial examination of the behavioral network for a particular application was the discovery of extremely wide distributions approximated by power laws for degree, strength, and weight for both Web clients and servers. The basic properties of these distributions are summarized for reference in Table 3.2.

The lack of any characteristic scale in these distributions was surprising to me and represented an intriguing challenge for further research. My initial goal in taking a graph-oriented view of network traffic had been to use these distributions as a way of uncovering anomalies, but the unbounded variance and (in some cases) lack of any well-defined mean imply that no amount of activity on the part of a particular Web client or server can be considered so outlandish that it will not arise in the course of normal network activity. It is worth noting that there were no *known* large-scale network attacks taking place across Internet2 during the period of data collection, but the width and extreme heterogeneity of these distributions imply this may be difficult knowledge to come by. If we attempt to detect anomalies in Web traffic crossing a transit network by setting up thresholds, we are left with no justification for the values of those thresholds other than our own

subjective impression of what seems to be excessive. The variance in legitimate traffic guarantees that some of it will spill over the threshold, yielding a constant stream of false positives.

The interplay between degree and strength was also surprising, especially the super-linear relationship between degree and strength in the case of Web clients. The notion that activity per connection can increase more quickly than the number of connection seems counter-intuitive: we would reasonably expect Web users of finite patience, reading ability, and time to interact with each server $less$, not more.

These findings thus became a primary motivation for a more comprehensive study of behavioral networks derived from Internet2 network flow data, which I describe in the remainder of this chapter.

3.3 Extended study

While the initial findings from the study of the Web behavioral network were both surprising and intriguing, they immediately bring to mind a number of questions. Are the scale-free properties observed particular to the Web, or are they a trait of a broad spectrum of network applications? Do they arise from some peculiarity of the primarily academic traffic found on Internet2 at that time, or can the results be generalized to the Internet as a whole? Assuming that these distributions are consistently well-approximated by power laws from day to day, are their slopes constant, or do they follow a clear trend as Internet traffic continues to grow and evolve? Because I had gathered and analyzed several other days of data in the process of validating the preliminary study, I had good evidence that the scale-free distributions were indeed a consistent phenomenon and that their slopes did not vary appreciably in the short term, but the rest of these questions remained unanswered.

A further study to explore these questions would be necessity have several properties, each of which can be satisfied with Internet2 flow data. First, it must expand the focus of the previous work and consider a full spectrum of network applications rather than just the Web. The flow data readily support segregating flows based on the origin application, given continuing trust in the heuristic that applications generally run on their well-known ports. Second, it should explore whether the results are strongly colored by the non-commercial nature of traffic on Internet2. I initially thought this would be an extremely difficult desire to fulfill, but the expansion of Internet2 to include peering relationships with a variety of commercial ISPs means that more recent samples of flow data ought to be much more representative of Internet traffic as a whole. Finally, it must include a longitudinal element so that we can see which of the properties discovered remain constant and which evolve over time. Thankfully, the structure of Internet2 and its measurement infrastructure have remained nearly constant over a span of several years; flows are still available in the same format, based on the same sampling rate, from a network with very similar structure.

Given the elusive nature of anomaly detection in the context of the Web behavioral network, I also felt it important to demonstrate that graph-based analysis of behavioral networks had some concrete predictive value beyond basic characterization for future efforts in capacity planning and modeling. The hope was thus that a more detailed study could yield a practical application of these analytical techniques, one that could provide useful information unavailable through the straightforward relational view of flow data and that would operate even when working with anonymized flow records and no access to payload data.

The study described in the remainder of this section fulfills all of these goals. This analysis is based on data sets B and C, which both comprise twenty-four hours of traffic gathered on April 14, 2005, and April 22, 2008, respectively. These were once again typical days in the life of the network, with no known major outages or disruptions of service. As was the case with the preliminary

study, the same analysis was conducted on several days around each of the target dates to reduce the chance that the data represent mostly transient phenomena. In both cases, the data appear to be genuinely representative of typical network conditions.

The remainder of this section is arranged as follows. In the first subsection, I present a structural analysis of data sets B and C equivalent to that performed in the preliminary study. This analysis is significantly more detailed than the previous work, as it includes consideration of several classes of application traffic and evaluation of trends observed between the two data sets. I also present some fragmentary results related to an attempt to extend the analysis to the distributions of clustering coefficients and spectral analysis. In Subsection 3.3, I describe how the behavioral networks for individual TCP ports can be used as a practical means of identifying and classifying network applications based only on their observed patterns of behavior, without any recourse to packet analysis.

Structural properties

There are, of course, many possible ways in which to divide the network flow data into categories based on the underlying application. In theory, each of the 65,536 possible TCP ports, and equally as many UDP ports, may represent a distinct network application. Because of the heuristic that classifies the endpoints of each flow as either clients or servers, we do not suspect many of them of hosting any application, but the list of ports that occasionally host services is long enough so that it is both impractical and taxing to the reader to consider each one separately. Fortunately, the way in which our use of the Internet has evolved over time provides us with an obvious first-level categorization of traffic. Put bluntly, there are only three varieties of application traffic on the Internet that occur in appreciable volume: the Web, peer-to-peer (P2P) applications, and everything else. Though a wide variety of applications from e-mail to network news to massively

multiplayer online games are thus thrust into the "other" category, the traffic associated with any of these applications taken individually pales next to the Web and P2P applications. The analysis in this section therefore focuses on these three broad categories.

It is worth noting that P2P applications, because of the often questionable nature of the activities they support, are difficult to isolate as a set of ports. Many users purposefully configure their own applications to run on non-standard ports so as to circumvent naïve attempts at filtering and traffic shaping. However, examination of the data show clearly that a great number of users *do not* do so, allowing us to get a reasonable overview of P2P traffic from a fairly small set of ports. In particular, we try to isolate the network flows associated with Bittorrent, Gnutella, eDonkey, Hotline, MSN, Napster, DirectPlay, IRC, and Kazaa.

Finally, these data sets also use a more encompassing definition of Web-related traffic. Instead of using only flows involving port 80, all flows involving ports 80 (HTTP), 443 (HTTPS), and 8080 (alternate HTTP) are considered to be Web-related.

Data set B, gathered in 2005, consists of over 600 million flows involving almost 15 million hosts. Of these flows, 258 million (41.3%) were Web-related, and an additional 82 million (13.1%) were associated with known P2P applications as defined above. While we must bear in mind that these classifications are based entirely on TCP port numbers and are thus individually suspect, the large and varied user population of Internet2 strongly implies that a majority of flows are correctly identified in this way. The remaining 285 million (45.6%) flows include all other traffic, which includes everything from network performance testing (a constant feature of Internet2 traffic) to e-mail and interactive logins.

Data set C, gathered in 2008, is substantially larger and contains over 980 million flows involving just over 18 million hosts. The proportion of flows and edges associated with the application

categories did shift over time, as shown in Figure 3.16. In particular, the proportion of flows associated with the Web appears to have grown, while that associated with P2P applications seems to have shrunk. However, this does not indicate an actual decline in the popularity of P2P applications, but rather evolution in the relative popularity of different P2P networks and the ports they use. In Subsection 3.3, I describe how application classification techniques make it possible to identify new ports that have become associated with Bittorrent. With the addition of these ports, the volume of traffic attributable to P2P networks is actually much greater in 2008 than in 2005. For this reason, the rest of the analysis of data set C includes these ports, which carry a relatively small number of very large flows that substantially increase the proportion of traffic associated with P2P applications.

In 2005, 5.82 million of the observed hosts behaved as clients and nearly twice as many, 11.1 million, behaved as servers, according to the more-frequent-port heuristic. Such a high proportion of servers to clients is an indication of scanning traffic on the network: when rogue clients are searching for vulnerable servers, they generate many probe flows to well-known server ports on hosts that may or may not actually exist. In the case of the Web and P2P applications, we observe the opposite effect. When we consider only Web-related flow, we find 3.97 million hosts behaving as clients and 0.68 million (less than one-fifth as many) behaving as servers. Similarly, for P2P traffic, we find 710,000 clients, but only 140,000 servers. The remaining flows describe the activity of 2.48 million clients and 10.6 million servers.

The bipartite behavioral network containing all hosts and applications is extremely large, containing 131 million edges. If we examine the subgraphs related only the particular classes of application (the behavioral networks for the Web, P2P, and "other"), we find that the Web graph contains 50.1 million edges (38.0% as many as the full graphs, the P2P graph contains 7.89 million edges (6.0%), and the remaining TCP traffic contains 54.9 million edges (41.6%). This is depicted

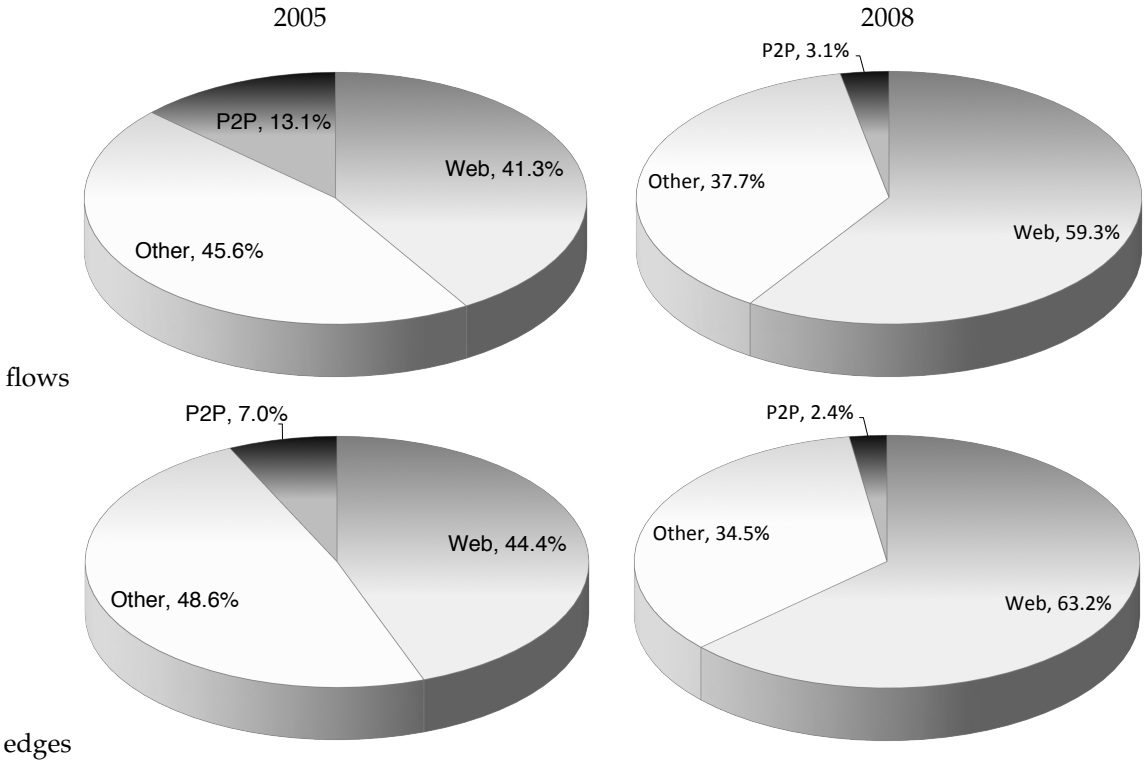


Figure 3.16: Top: Proportion of collection flows generated by each category of traffic in 2005 (left) and in 2008 (right). Bottom: Relative sizes of the bipartite graphs for each category of traffic, as measured by the number of edges, in 2005 (left) and in 2008 (right). As mentioned in the text, these charts underestimate the contribution of P2P applications to the 2008 data set.

graphically in the lower half of Figure 3.16.

For each category of traffic, it is illuminating to examine the degree of overlap between the sets of client hosts C and server hosts S, represented with the quantity

$$O = \frac{|C| \cap |S|}{|C| \cup |S|}$$

. When $O = 0$, we have no host acting as both client and server; when $O = 1$, every host acts in both roles. We would expect O to be lower for traditional client-server applications than modern P2P applications: few NNTP servers also have local users who read news, but it is a basic expectation of the protocol that Bittorrent users will both make and receive connections as they exchange data. Indeed, we find that $O = 0.013$ for Web traffic, as compared to 0.097 for P2P traffic. This is a strong indication that hosting content for the Web is much less of a participant sport than sharing personal files, in that relatively few users run both browsers and Web servers. However, it must be noted that $O = 0.15$ for other traffic, which suggests the presence of significant amount of covert P2P traffic within this unclassified data. Subsection 3.3 describes a method with potential utility for classifying these flows irrespective of whether their TCP port numbers are known.

The 2008 data set indicates a much smaller amount of scanning traffic, with clients outnumbering servers by a healthy margin (12.8 million to 9.0 million). It replicates the overwhelming dominance of clients and servers in the categories of Web and P2P traffic (6.0 million to 940 thousand, and 540 thousand to 160 thousand, respectively). The small degree of overlap between C and S for Web traffic is still present, with $O \simeq 0.01$. Somewhat surprisingly, the overlap declined by more than half for P2P traffic, with $O \simeq 0.04$. One possible explanation for this low degree of overlap may be that the regulatory hazards of file sharing have encourages many participants in P2P networks to operate in bad faith: the fact that the ratio of clients to severs is over three to

Table 3.3: Volume of TCP traffic observed for major classes of network application as determined by TCP port number. As discussed in the text, the port assignments for P2P applications in the 2008 data set include emergent ports not represented in the 2005 data.

	Web		P2P		Other	
	2005	2008	2005	2008	2005	2008
Proportion of traffic	17.4%	28.5%	4.0%	7.1%	78.6%	64.4%
Mean data (client)	81 kB	146 kB	105 kB	395 kB	586 kB	250 kB
Mean data (server)	471 kB	936 kB	515 kB	1270 kB	137 kB	243 kB

one strongly implies that many users prefer downloading files to providing them to other. Other possible causes include the rise of projects that simply monitor the content of P2P networks and asymmetries introduced to the flow data by failed connections to servers that are no longer participating in Bittorrent file swarms.

The total volume of traffic recorded in 2005 was approximately 1.85 terabytes, with a mean of 124 kB per host. In 2008, the figure had risen to 6.18 trillion bytes, with a mean of 343 kB per host. We must remember that sampling causes a strong underestimate in these figures, but the stability of the network structure over time makes the means directly comparable. Table 3.3 shows the results of breaking this traffic down into the same broad categories of application as before. An important note here is that the values for "other" traffic are influenced by large volumes of *iperf* test traffic generated by the Internet2 Network Operations Center for performance testing, which may exaggerate this category relative to other major transit networks. Figure 3.17 offers a visual comparison of the proportions of traffic by category in 2005 and 2008.

These statistics offer only a high-level description of some basic features and trends of network traffic as a whole; they tell us little about the role a typical user actually plays in the network. The focus of this analysis thus turns to the structure of the individual behavioral networks corresponding to the Web, P2P applications, and everything else.

As in the preliminary study, we begin by considering the distributions of degree and strength for

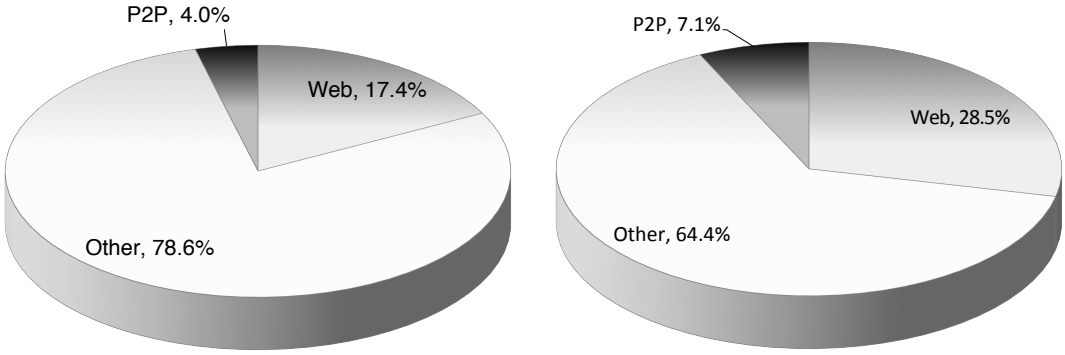


Figure 3.17: Proportion of traffic volume consumed by each category of traffic in 2005 (left) and in 2008 (right). Again, as discussed in the text, the port assignment for P2P applications in the 2008 data set incorporate additional ports not used by the P2P community in 2005.

the nodes in each behavioral network. Recall that the degree of the node in each behavioral network reflects the total *count* of users with which it has exchanged data, and the strength reflects the total *amount* of data it has exchanged. Furthermore, these various behavioral networks are constructed simply by aggregating traffic by specific ports, so that the identities of nodes are consistent across all three behavioral networks.

Because both the degree and strength distributions reflect the decisions made by a large population of individual users, it might seem plausible for their form to be roughly normal. However, we have already observed in the preliminary study that this is not true of Web traffic. In Figure 3.3, we can see the effect is not limited to the Web. All of the degree and strength distributions shown have extremely long tails, some spanning almost ten orders of magnitude. As an example, the mean strength $\langle s_C \rangle$ of a client in 2005 was approximately 318 kB, but the standard deviation of the distribution was 72.6 MB: once again, we have a situation in which the level of statistical fluctuation is over two orders of magnitude greater than the mean value.

In the case of all traffic and the Web behavioral network, we are able to approximate both the degree and strength distributions with the now familiar power-law approximations $P(k) \sim k^{-\gamma}$ and $P(s) \sim s^{-\gamma}$ over several orders of magnitude. We find fits for γ of roughly 2.4 for client degree,

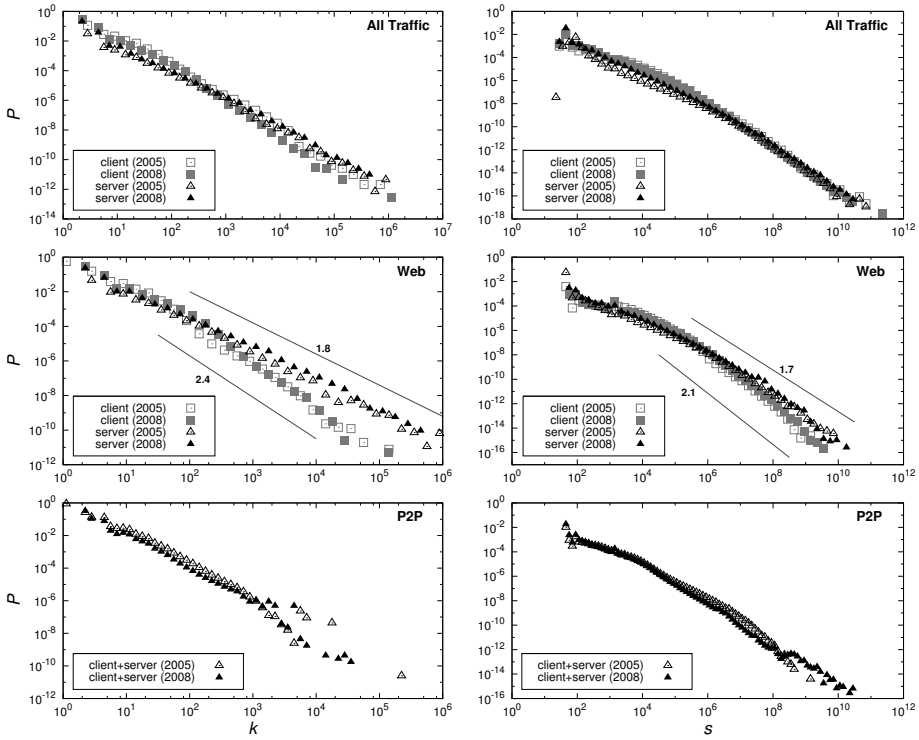


Figure 3.18: Probability distributions for degree k (left) and strength s (right) in the Internet2 behavioral network in 2005 and 2008, shown for all data (top), the Web (middle) and P2P applications (bottom). The annotated lines in the Web plots show statistically significant best-fit power-law approximations to the actual data, with $R^2 \geq 0.995$.

2.1 for client strength, 1.8 for server degree, and 1.7 for server strength. Remarkably, these properties are consistent with both the results of the preliminary study and between the two data sets. We observe little change in these exponents even after three years of rapid change in the network itself, including the introduction of large volumes of commercial traffic. The strong implication is that these exponents describe not just a snapshot of a typical day on this particular network, but rather inherent properties of the behavior of a large user population in the Internet.

The importance of the slopes of these power-law approximations cannot be over-emphasized. Recall from the preliminary study that we have two critical regimes of interest.

When $\gamma < 3$, the second moment of the random variable x, given by

$$\langle x^2 \rangle = \int x^2 P(x) dx$$

, diverges, implying that the standard deviation is not an intrinsic value of the distribution and is bounded only by the size of the data sample. This makes the mean value $\langle x \rangle$ no longer typical, putting us into the world of "scale-free" behavior, in which there is an appreciable probability of finding a client that has contacted any arbitrary number of servers or downloaded any arbitrary amount of data. The mean values of degree and strength appear to be of no value is predicting user behavior.

When $\gamma < 2$, as continues to be the case for Web servers, we have the even more dramatic situation first encountered in the preliminary study. In this case, even the first moment

$$\langle x \rangle = \int x P(x) p x$$

diverges and is bounded only by the size of the sample. Neither the mean number of connections nor the mean amount of data exchanged are intrinsic to the system, and we can say nothing

meaningful about a "typical" Web server for any of the data sets gathered.

In the case of P2P networks, although we continue observe heavy-tailed distributions, there appears to be a definite exponential cutoff. Beyond this cutoff, the probability functions decay more quickly than a power-law fit would predict. One possible explanation for this is that hosts participating in a P2P network are far more likely than other types of server to be personal computers. The limited computing and network capacity of these smaller components may account for this exponential decay, a notion supported by the increased width of the 2008 distribution: as the power of individual workstations increases, the cutoff can move to the right.

The interaction between degree and strength is still of interest, particularly for comparing the nature of this interaction across the application classes and over time. Again, because of the power-law nature of the distributions of degree and strength considered separately, it is unsurprising to observe strength increasing as a function of degree and for the relationship itself to follow a power law. These relationships are illustrated in Figure 3.3.

Recall from the preliminary study that although we can expect basic power-law behavior $\langle s(k) \rangle \sim k^{\beta}$ of the interaction between degree and strength, it is the value of the exponent β that is of greatest interest. If the approximation is good, then $\beta < 1$ implies a sublinear relationship, with diminishing traffic per connection as the number of connections increases; $\beta = 1$ implies a trivial linear relationship; and $\beta > 1$ implies a super-linear relationship in which traffic per connection increases exponentially as the number of connections increases. In the case of server behavior, we observe a linear or sublinear relationship ($\beta \leq 1$), but in the case of Web clients, we continue to observe $\beta = 1.2 \pm 0.1$, giving clear indication of a persistent superlinear relationship consistent across all three data sets. The transition from academic to commercial traffic has not altered this effect, suggesting that it too is an inherent feature of Web clients in general.

Finally, I consider the assortativity of the behavioral networks for the 2008 data set. Recall that

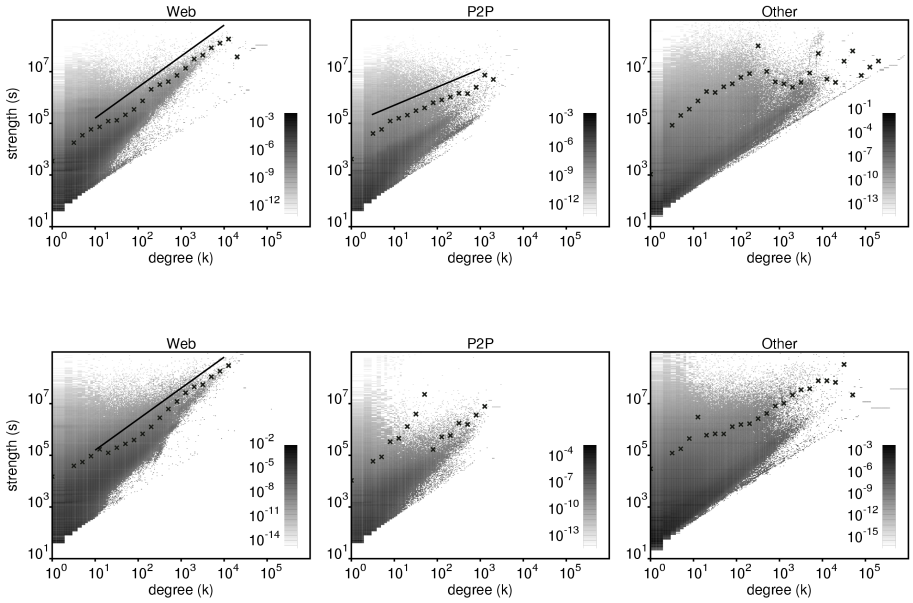


Figure 3.19: Behavior of strength (total data) s as a function of degree (number of hosts contacted) k for the Web, P2P, and other traffic, in 2005 (top) and in 2008 (bottom). As in the earlier density plots, the tones represent the frequencies of strength values, normalized within each degree bin, on a log scale. The plotted points show the mean strength of degree bins, and the lines serve as a visual reference to power-law approximations of the actual data. For both data sets, the behavior of Web traffic is roughly linear on a double logarithmic scale.

assortativity is a measure of whether high-degree nodes in a network connect to other high-degree nodes (high assortativity) or to low-degree nodes (disassortativity). Because of the bipartite nature of the data, I consider the assortativity of the CTOS (client-to-server) and STOC (server-to-client) connectivity matrices separately. In both cases, I first find the degrees of the source and destination nodes for each edge. Then I plot the mean degree of destination nodes as a function of source degree, using appropriately normalized logarithmic bins for the source degree. From the CTOS matrix, we thus obtain a view of whether clients of high out-degree contact servers of high in-degree; and from the STOC matrix, we find whether servers of high out-degree are contacted by clients of high in-degree. The same procedure can be applied to strength, in which case we find out about aggregate traffic rather than the number of contacts.

The results are shown in Figure 3.3. In the case of degree (top), we can clearly see that Internet traffic in general, and the Web in particular, are strongly disassortative, which is generally consistent with graphs derived from technological designs [120]. This effect is largely absent from the P2P behavioral network, whose structure is much more determined by social interaction. Indeed, when we examine strength (bottom), which is much more closely analogous to user behavior than degree, we find that the dissortativity of All and Web traffic is markedly reduced, and that P2P traffic is actually assortative.

Unfortunately, this study does not incorporate examination of the distribution of the clustering coefficients of the nodes of the behavioral networks, for several reasons. The most immediate is the computational complexity involved in calculating the distribution. The degree distributions of the behavioral networks are so broad that the most extreme nodes have a degree on the order of the size of the network itself, making the running time of calculating the clustering coefficient of those nodes $O(N^2)$. There is also some difficulty in applying the standard definitions of the clustering coefficient to a bipartite network: we would naturally expect that the servers contacted

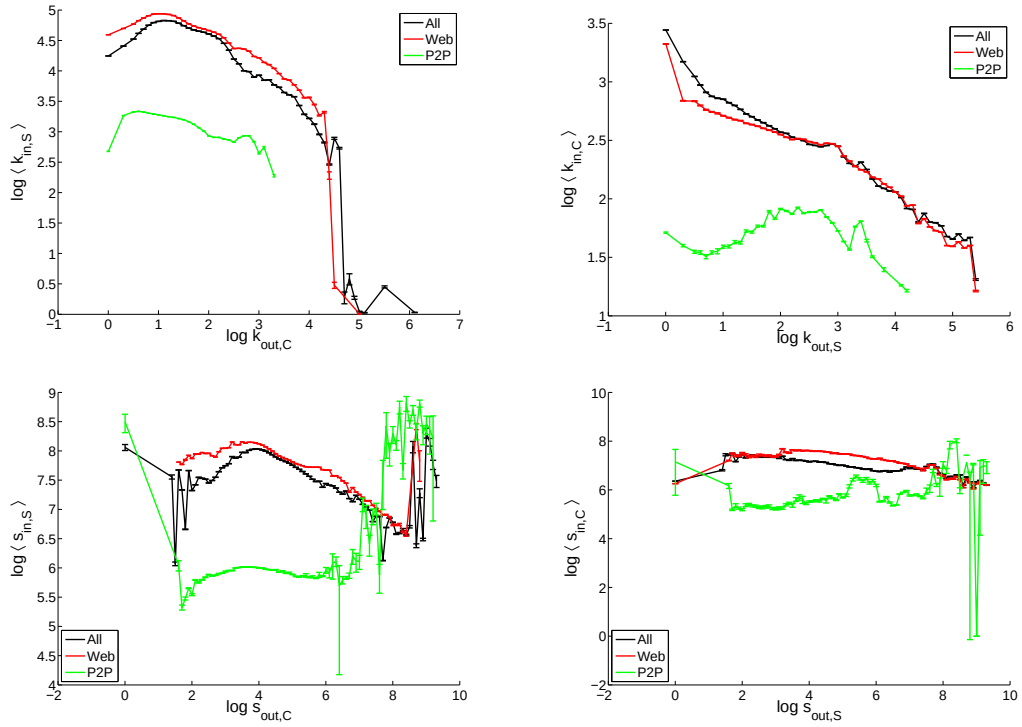


Figure 3.20: Assortativity of degree (top) and strength (bottom) in the 2008 behavioral networks. On the left, in-degree [in-strength] of servers is shown as a function of the out-degree [out-strength] of the clients contacting them. On the right, in-degree [in-strength] of clients is shown as a function of the out-degree [out-strength] of the servers they contacted. In each graph, the independent variable is logarithmically binned, and the error bars indicate plus or minus one standard deviation of the mean.

by a Web client not to have much direct contact with one another. An appropriate approach for future research would be to examine clustering in unipartite client and server co-citation networks derived from the behavioral networks.

I have also omitted results from spectral analysis here, for a lack of substantive and interpretable results. My experience thus far in the application of eigenvector analysis to large scale-free networks has been that the values of the eigenvectors themselves obey power-law distributions over many orders of magnitude. Plotting principal eigenvectors against each other fails to reveal clear anomalies, and applying them in standard ways to clustering techniques does not yield clearly identifiable clusters. I elaborate more on these problems in Section refsec:click-host.

The findings of this study carry a variety of implications. Primary among them is that the heterogeneity seen in the degree and strength distributions of Web servers, fitting a power-law distribution with $\gamma < 2$, is so extreme as to make their mean values no longer well-defined. This behavior is consistent across several years of network traffic, strongly suggesting that this is an intrinsic feature of the Internet rather than a chance observation. This means that we can infer no global average quantity for either the number of clients or the amount of data transmitted for a Web server. Not only does this imply that no single scale is most appropriate for the design of general-purpose Web server software, but it also means that we cannot design effective security systems for transit networks that examine questions such as "which Web servers are receiving a disproportionate number of hits?" Any answer based on simple thresholds will include many false positives: for any virtually any amount of traffic, there are servers that legitimately receive that much traffic.

As mentioned above, the existence of the exponential cutoff in the degree and strength distributions for the P2P networks may be a results of the limited processing power and network capacity of individual workstations. The data seem to bear out this notion, as witnessed by the lengthening

distribution over time. Given that the desire of users to download data appears to be unbounded, especially because individual files exchanged become larger and larger as video standards evolve, the movement of these distributions can be used as a rough measure of the pace at which the power of the average computer is improving.

The superlinear relationship between strength and degree for Web clients, first observed in the preliminary study and confirmed by both data sets in the second study, suggests that the amount of data exchanged with each Web server tends to increase as a user contacts more servers: the more sites surfed, the more data is received from each of those sites. Such a non-linear growth mechanism may be the basis for techniques to disambiguate the behavior of individual Web surfers and large-scale crawlers. Individual users clearly lack the ability to digest an ever-growing body of information from each site as they surf the Web, whereas Web crawlers are designed to do exactly that; these behavior may form identifiable clusters. I have not explored this hypothesis in the present work, but it could be tested given access to Web click data of the type used in Chapter 4 from a transit network.

The relationship between the in- and out-distributions of strength may also facilitate the discovery of unrestricted proxy servers that are the launch points for a wide variety of security attacks on the Internet. Most of the traffic associated with a proxy for an application will be repeated: requests from a client to the proxy are retransmitted from the proxy to a server, and the server response is likewise retransmitted by the proxy back to the client. We would thus expect a proxy to exhibit an unusually high level of symmetry in its role in a behavioral network; not only would it function as both client and server, but its in-strength and out-strength would be nearly equal to one another. This avenue of research would require close coordination with the Internet2 security community and is also for future consideration. It should also be noted that the removal of open proxies is not an unmitigated good: besides facilitating attacks, they also serve as a key means of

access to the greater Internet for users in countries with onerous censorship restrictions. The abolition of open proxies could have the effect of reducing fairly benign Internet attacks while strongly compromising the ability of many to obtain uncensored information from the outside world.

Application classification

The results reported thus far are illuminating, novel, and have important implications for modeling and understanding Internet traffic. However, most of these results are necessarily couched in negative terms: we cannot speak of an average host, we cannot do simple threshold-based anomaly detection, and so forth. Under the circumstances, it is natural to ask what positive results can stem from the use of anonymized network flow data. Are there meaningful questions that we can answer using this data source, and better yet, answer more confidently than we could through more traditional means of analysis?

This section describes a tested application of anonymized flow data in which distributions from behavioral networks projected onto ports. This makes it possible to construct an application network in which the nodes are network applications rather than hosts, and the edges are measures of behavioral similarity among those applications. I then apply hierarchical clustering to the nodes of this application network to create a taxonomy of applications organized by their observed behavior. The techniques described here yield a system that can both classify network applications in a robust way based on their behavioral characteristics and identify novel ones, all without access to the payload of a single packet.

There is ample motivation for such an effort: the large volume of data in the "everything else" behavioral network serves as a vivid illustration of how non-standard applications comprise a substantial portion of Internet traffic. As I have mentioned before, researchers and system administrators often have a fragmentary and incomplete knowledge of what is actually taking place

in the cyberworld, even in the fact of increasing legal and ethical demands that they categorize and monitor application traffic. The central problem is inherent to the design of the Internet protocols themselves: ports have no formal semantic meaning other than as a numeric label using for multiplexing at the transport layer. Their association with specific applications is often quite strong—after all, we seldom see a URL with an explicit port identification—and has been the basis of the flow analysis discussed so far, but amounts in the end to nothing more than custom.

Assuming the standard pairings between ports and applications to be true does yield accurate classification for a large segment of user activity, but there is a growing need to consider the proportion of Internet traffic generated by applications running on non-standard ports or covert channels. Many users evade local firewall policies by running P2P application on ports normally associated with the Web. Evading custom in this way gives users a variety of ways to disguise their activity, including serious misconduct such as operating spam relays, managing networks of compromised hosts, and so forth. This traffic is not all easy to discern from the rest; most network security systems can be evaded through encryption, packet fragmentation, port-knocking, and a variety of other techniques. The consequence is that while we can monitor and be aware of the *existence* of applications that mediate some sort of interaction among user, we often do not know what *kind* of communication and function they support. Flows do not explicitly encode intent or purpose.

The behavioral networks described in this chapter have direct utility for the problem of identifying applications. Let us begin by considering the nodes in the behavioral network associated with a particular port p. Because servers are likely to be devoted to hosting a single application and are much less likely than clients to represent the actions of a single user, we will restrict our attention to the set of client nodes C. We define the *port strength* of a client $i \in C$ as being equal its

out-strength

$$s_i^p = \sum_{j \in C \cup S} w_{ij}^p$$

. This notion of port strength reflects the total amount of data client i has exchanged using port p. The motivation for focusing on out-strength is simply that many connections fail, and a client's attempts to establish communication are often as informative as their actual connections.

We are thus able to construct a vector of port strengths for each application port p represented in the flow data:

$$\vec{p} = (s_1^p), \ldots, s_{|C|}^p)$$

. Recall that the node labels are consistent across all behavioral networks, so that the size of each port strength vector is the same and their corresponding elements represent the same host. This suggests that we can measure the correlation of use between two ports p and q as a function of their port strength vectors $\vec{p}$ and $\vec{q}$. One simple and appropriate measure is the cosine similarity of the vectors, given by

$$\sigma(\vec{p}, \vec{q}) = (\vec{p} \cdot \vec{q})/(\|\vec{p}\| \cdot \|\vec{q}\|)$$

. This measure ranges from zero in the case of completely orthogonal use, to one in the case in which every host uses the two applications to the same proportion.

This measure of correlation of use allows us to build a matrix S in which each entry S_{pq} is the cosine similarity of $\vec{p}$ and $\vec{q}$. We can interpret this matrix as being the weighted connectivity matrix for an undirected *application network*. This is a fully connected graph having ports as nodes and strength correlations as weights. The underlying intuitions behind such a representation are that applications used in similar ways are likely to have similar functions, and that enough people use related applications within a category to make port strength vectors a viable method of quantifying similarity of usage. Once we have constructed this application network, it becomes possible to

construct a taxonomy of ports based on their observed behavior.

To test these ideas in a concrete way, I generated the behavioral networks and associated port strength vectors of the 36 TCP ports with highest traffic by volume in the 2005 data set. After building the similarity matrix based on pairwise comparison of these ports, I converted each cosine similarity value σ into a distance measure d using the relation $d = (1/\sigma) - 1$. Interpreting the similarity as a distance makes it possible to apply Ward's hierarchical clustering algorithm to the data, yielding a binary tree that represents a taxonomy of the ports based on their observed traffic [146]. This algorithm is not uniquely well-suited for the purpose; other clustering algorithms, such as *k-means*, yield similar results.

Figure 3.3 illustrates the results of applying this procedure. The dendrogram corresponding to the results of the hierarchical clustering is displayed along the edge of the matrix, with the shading of the cells themselves based on the logarithms of the similarity values, with a cut-off at $\sigma = 10^{-4}$. The matrix has been reordered by hand consistent with the structure of the dendrogram to call attention to the natural groupings that emerge from the clustering. We can see clearly that the Web is so pervasive among users so as to be strongly correlated with virtually every network application; moreover, as one might expect, the various ports associated with the Web are also tightly clustered with one another. The groupings of the remaining applications also capture their function. Bittorrent, currently the most popular P2P application on the Internet, uses a variety of different ports, but they form a tight cluster. Another strong cluster identifies the ports used by the network for Gnutella, the next most popular file sharing application. Standard client-server applications also form clusters. One includes ports used by email, chat, and file transfer protocols; another includes applications for listening to streaming music and logging into work from home. Several other P2P applications are also clustered together.

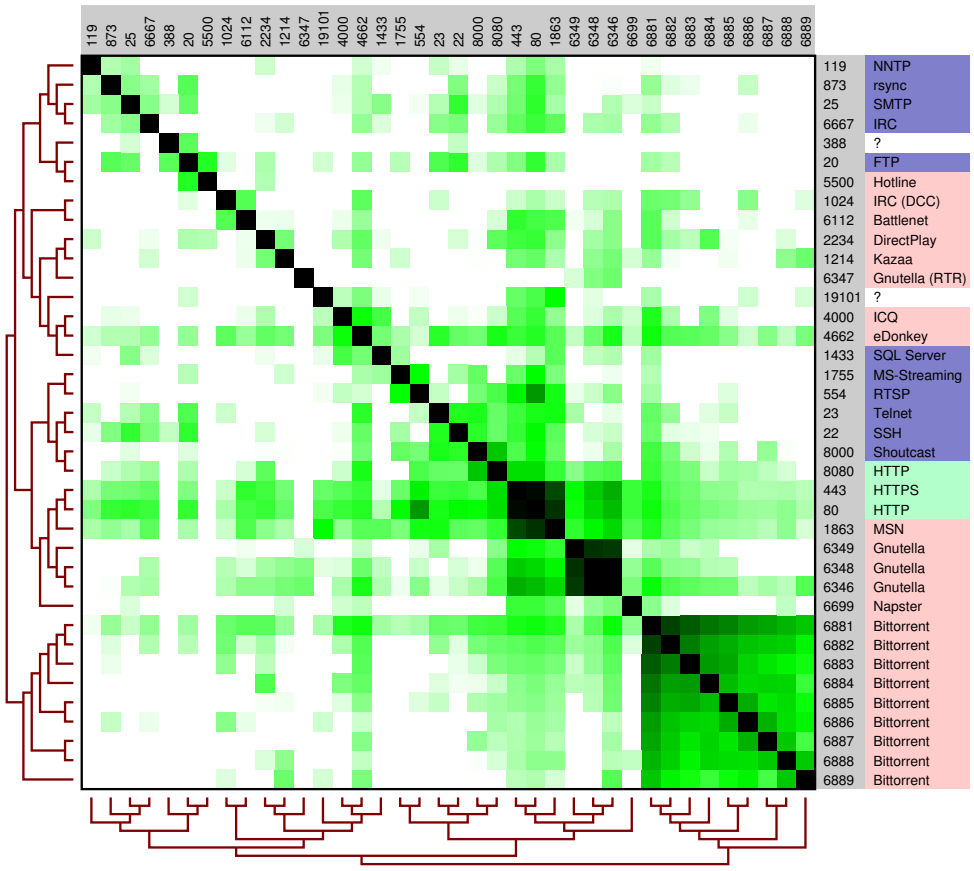


Figure 3.21: Correlation of client use of network applications as measured by cosine similarity of port strength vectors in the 2005 data set. The matrix contains the 38 ports with highest traffic, two of which are used by unknown applications, as explained in the text. This is a symmetric matrix showing the log-scale correlation between each pair of ports (with a threshold of 10^{-4}. The dendrogram used to organize the sorts is obtained using Ward's hierarchical clustering algorithm [146]. The ports have been manually labeled and colored according to three broad classes: P2P (pink), Web (green), and traditional client-server (blue) applications. A different color is used for ports associated with the Web because of their ubiquity and frequent use in conjunction with other application, e.g., file transfers in Gnutella.

While this clustering mirrors a general understanding of network application, it is of little practical value unless it can be used to predict the nature of previously unknown applications. To explore the utility of the clustering processes, I extracted the behavioral networks for an additional sixteen TCP ports whose primary function was unknown, either because of the obscurity of their assignment or because they had not been formally assigned at all. The procedure for predicting the function on an unknown port is straightforward: simply calculate the similarity of its port strength vector to the known ports, repeat the clustering processes, and examine the location of the port in the resulting hierarchy.

Two of the unknown applications, represented by a '?' next to their port numbers, are included in Figure 3.3. We can see the port 388 is couple most strongly with FTP and Hotline (an older P2P file-sharing application that has faded from use). Subsequent investigation showed this port did indeed have a formal assignment to "Unidata/LDM," a file transfer application used for moving large sets of meteorological data between research centers. In other words, the function of the application is indeed to perform file transfers in a strongly connected peer-to-peer network.

The results for port 19101 are even more compelling. This port was grouped with both instant messaging and P2P applications, suggesting that it might be a P2P application that relies on individual contact among users to initiate file transfers. This prediction provided enough context to construct search engine queries to discover (with the additional assistance of Google's translation service) that this port is used by "Clubbox," a Korean file-sharing program that allows users to trade entire seasons of television programs on large virtual hard drives. The application includes a interactive chat facility through which users mediate their sharing. In this case, the prediction was not only correct, but sniffing the application data would have been of little use to a network engineer unfamiliar with the Korean language; the application network offers information that packet analysis alone does not.

This example highlights a central contribution of this research: because the data source consists of flows rather than packets, the methods presented here are not only more respectful of user privacy, but they do not depend on our ability to interpret the payloads of individual packets. While it is possible that the right network engineer might be able to recognize the Korean language in the port 19101 packets, the same is not true if these packets had been encrypted. The application classification techniques described here continue to perform robustly even if the entirety of the data streams are encrypted and their contents impossible to decipher.

The predictions based on clustering and the actual identities of the applications for all sixteen unknown applications are shown in Table 3.4. These predictions are based entirely on the location of the application within the port taxonomy rather than external domain knowledge. I tested the predictions by using them in conjunction with my own experience in network administration and security to perform directed Web searches. Of the sixteen predictions, eight were entirely successful. A further six predictions were partial, and the remaining two could not be verified or disproved. The partial predictions result from applications that were clustered with both P2P file-sharing applications and ports strongly associated with malicious activity (IRC and SQL Server). In these cases, there was insufficient data to make a judgment as to which of these purposes was the more likely. In a practical application of these techniques, network administrators would be advised to examine such cases especially closely, as the ambiguity is also an indication that systems involved with P2P applications are more likely to be compromised by malicious software, possibly through the P2P applications themselves. The predictions for two of the ports could not be verified because they were in use only transiently during the data collection period and no longer carry more traffic than any other randomly selected port. The window of opportunity for classifying those ports had simply passed.

To clear a possible point of confusion, the prediction of P2P file transfer for a port that turned

Table 3.4: Predicted uses of high-traffic TCP ports in the 2005 data set running lesser-known applications, based on the hierarchical clustering data; and their actual uses, as derived from security bulletins, Web searches, application home pages, etc. The ports marked as "unknown" were in use only transiently and did not carry appreciable traffic on other days.

Port	Predicted	Actual	Match?
388	traditional file transfer	weather data transfer	yes
19101	P2P chat or file transfer	individual file shares	yes
9080	P2P with central index	team collaboration	yes
8090	Windows P2P w/ Web svc.	Weblog server	yes
5020	Windows P2P file transfer	BBFTP file transfer	partial
42899	P2P file sharing or trojan	*(unknown)*	unknown
8301	P2P file sharing or trojan	several trojans	partial
1025	trojan	many different trojans	yes
20000	P2P, probably BitTorrent	BitTorrent	yes
59174	P2P file sharing or trojan	*(unknown)*	unknown
20001	P2P file sharing or trojan	several trojans	partial
15002	P2P file sharing or trojan	biology collab. tool	partial
16881	P2P, probably BitTorrent	BitTorrent	yes
9000	P2P file sharing or trojan	several trojans	partial
3124	Windows P2P file transfer	Web proxy (Windows)	yes
39281	P2P file sharing or trojan	grid-based computing	partial

out to be involved with Web proxy traffic is indeed successful. While Web proxies predate P2P file-sharing networks, their actual function in the network is essentially that of a moderator between peers. A Web proxy draws data from a collection of information provides and then shares that same content with a community of users, making its traffic not only somewhat symmetric, but directly analogous to that of P2P applications that download a file (or parts of a file) and then share them with other users. Though they are not usually described as such, they are actually an early form of P2P application.

The Internet is a rapidly changing environment, and the popularity of network applications can change drastically in a brief period of time as users gravitate to newer technologies or court rulings mandate the death of older ones. The purpose of applications is equally dynamic: at one time, USENET was used primarily for the exchange of news and personal communication, and

its protocol (NNTP) was developed with those goals in mind. However, its traffic is now overwhelmingly dominated by people broadcasting media files to the larger community. Similarly, IRC was developed as a chat protocol, but is now used just as frequently as an enabling technology for peer-to-peer file transfers. A key strength of the approach described here is that applications are clustered based not on the design of their protocol, but the way in which they are being used. If the usage pattern of an application changes, its place in the application taxonomy will shift on its own without human intervention.

Because of this environment of rapid change, while this study illustrates the utility of the application network for the 2005 data set, such a demonstration is incomplete without applying it to more recent data as well. To better understand the pace and structure of the evolution of network applications, I applied the same clustering analysis to the 2008 data set, once again using the TCP ports with the highest volume of traffic. The results are shown in Figure 3.3. It is immediately clear from this analysis that between 2005 and 2008, several new Bittorrent clients had become popular, eclipsing the popularity of the original client, and that these clients use a different set of default ports (20000–20029 and 40000–40007). This is the motivation behind the previously mentioned revision of the set of ports associated with P2P applications in the 2008 data set.

Because the utility of clustering in identifying applications met with such success in the 2005 data set, I applied it to the 2008 data set a well. In this case, I selected fourteen more unknown applications and attempted to classify them based on their roles in the application network. These fourteen applications include port number that were also among the unknown applications in the 2005 data set. This may at first glance seem inappropriate, but the choice is justified by the even more rapid pace of change in the use of informally allocated ports: the 2005 guesses might well be no longer valid in 2008.

The results, shown in Table 3.5, reflect a similar level of success: there were ten successful

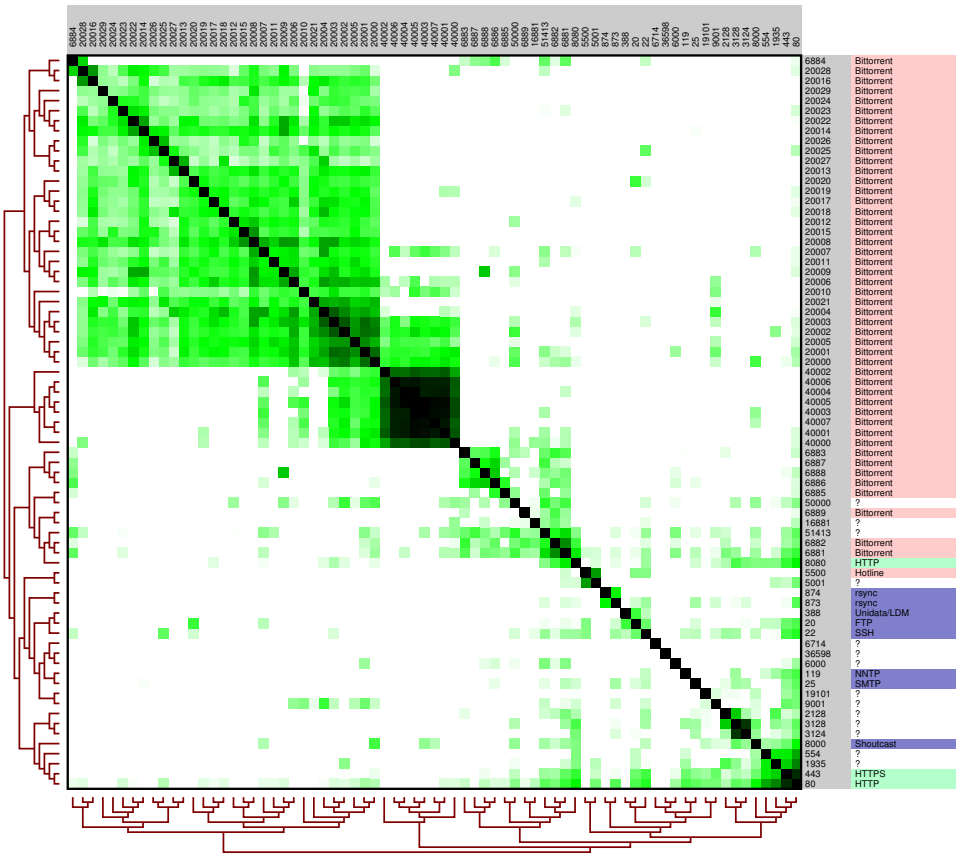


Figure 3.22: Clustering based on correlation of client use of network applications for the 2008 data set. The methodology and labeling are the same as in Figure 3.3. Also included are 14 unknown applications with no formal port assignment, whose purpose is deduced from their position in the application hierarchy.

Table 3.5: Predicted uses of high-traffic TCP ports in the 2008 data set, using the same methodology as in 2005.

Port	Predicted	Actual	Match?
50000	P2P, probably BitTorrent	BitTorrent	yes
16881	P2P, probably BitTorrent	BitTorrent	yes
51413	P2P, probably BitTorrent	BitTorrent	yes
5001	traditional file transfer	iperf and several trojans	partial
6714	traditional client/server	Internet Backplane Protocol	yes
36598	traditional client/server	*(unknown)*	unknown
6000	traditional client/server	X Window System	yes
19101	centralized file xfer svc.	file sharing (still ClubBox)	yes
9001	centralized file xfer svc.	Tor network	partial
2128	Web-related service	Net Steward dist. firewall	partial
3128	Web-related service	Squid Web cache	yes
3124	Web-related service	PlanetLab Web proxy network	yes
554	Web-related service	real-time streaming protocol	yes
1935	Web-related service	Macromedia Flash	yes

matches, three partial matches, and another instance of a port that was in transient use only. The partial match for port 5001 results from the dual use of this port for both *iperf* performance testing traffic and several well-known backdoor application. The Tor network cannot be regarded as better than a partial match because it is a general-purpose network application that provides anonymous transit for any type of data. While it is likely that much of the traffic carried by Tor involves sharing copyrighted media files, this is by design quite difficult to prove. The final partial match results from the guess that the Net Steward distributed firewall system was a Web-related service: while the firewall itself is not necessarily focused on the Web, the application does include a popular Web content filtering system.

The application clusters identified through this clustering technique show that system administrators or network managers can use graph-based flow analysis to infer traits of the activity carried out on a particular port, even if the application for that port is unknown, the data are encrypted, or the port is being used covertly. We are able to group applications together by the way they affect

the network rather than the origin of their code base or even their nominal purpose. This emphasis on how applications affect the network rather than their technical identity is of much greater utility for modeling user behavior and performing capacity planning. If we took it for granted that NNTP were still used primarily for its intended purpose of facilitating text-based conversations, we would scarcely think it worth examining. However, these results highlight that it is now used more for broadcasting movies and applications, and it must be dealt with on those terms. Because the clustering is based on aggregate behavior rather than pattern-matching against capture data, we can recognize the ports associated with new Bittorrent clients without ever examining the payload of a single packet. Moreover, the application clusters naturally evolve over time to reflect contemporary usage without a constant need for domain information from human experts.

3.4 Sampling bias

Because the findings presented in this chapter are both novel and somewhat dramatic—ideas such as the distribution of Web server traffic having no well-defined mean at all are difficult to understand on an intuitive level—careful attention must be paid to potential sources of bias within the data collection and analysis. There exists an entire industry based on threshold-based anomaly detection in network traffic, and it would be both unfair and unwise to recommend radical change based on effect that turn out to be a side-effect of experimental bias rather than features of actual Internet traffic. The aim of this section is to consider potential sources of bias in my analysis and consider how these may be discounted, quantified, or ameliorated.

Sources of bias

The greatest potential source of bias in this analysis is the fact that the flow records are generated based on a sample of the packets rather than the full flow of network data. Such sampling immediately raises a variety of questions: Are all packets equally likely to be sampled, regardless of any features of the packet, such as its size, endpoints, choice of ports, fragmentation status, etc.? Is each packet subject to a $p = 0.01$ change of being sampled, or is precisely every hundredth packet being sampled? Are router vendors able to confirm that the sampling takes place in exactly the way stated, or are they making optimistic assumptions about their own code?

Each of these concerns has some justification. The design of modern routers is such that most packets are forwarded directly by the individual line cards, without the general-purpose CPU becoming involved except in the case of "special" packets that require processing rare enough to be inappropriate to design into an Application-Specific Integrated Circuit (ASIC). Examples of such packets include those with a TTL field of one (i.e., those about to expire instead of being forwarded), those with the *record-route* IP header option set, and so forth. If these packets are not sampled with the same frequency as those handled entirely by the line cards, a systematic bias for or against these packets results. Whether packets are sampled independently or on a "every hundredth packet" basis may make no meaningful different during peak operating hours, but during periods of light traffic, flows with heavily periodic characteristics may be over- or under-sampled. The final concern, that of whether the vendors are sure of the properties of their own sampling methods, stems from the history of network management applications. Monitoring and measurement functions have long been low-priority features because, however important they may seem to the research community, they do not sell routers. It is not uncommon for serious bugs in management features to remain unpatched for several years in network operating systems from major vendors. One can reasonably question whether sampling code produced with a low priority can be trusted as fully as

the rest of the platform, particularly when it may be need to be updated in response to new product features.

In the event that packet sampling is performed on a completely even-handed basis, exactly as advertised, it is tempting to treat that as the final answer and assume that it makes no practical difference other than to underestimate the volume of traffic by a factor of a hundred. A moment's reflection, however, will show that this is not the case. First, the chance of a flow being represented in Netflow data is not linear in the number of packets it contains. A flow of a single packet has a $p = 1 - (1 - 0.01) = 0.01$ chance of being detected by a router, but a flow of two packets has a $p = 1 - (1 - 0.01)^2 = 0.0199$ chance of detection, rather than 0.02. Even a flow of a hundred packets has only a 63.4% chance of detection. This nonlinearity introduces a systematic bias toward very short flows that have a greater chance per packet of being detected. Second, when we receive a flow record with a single packet, we cannot know whether that represents one out of two, five, or ten actual packets. We can assign a maximum likelihood, but that requires making assumptions about the relative frequencies of flows of various lengths—information we can only obtain from the flow records themselves!

The complexity of the issues related to packet sampling motivated me to conduct an empirical study into real-world sampling bias, which is described at length in the ensuing subsections. However, before I introduce the details of this study, we must consider some other potential sources of bias in the flow data.

A frequent concern I have heard expressed about the use of Internet2 network flow data is whether the user population of Internet2 is sufficiently representative of the Internet as a whole for any findings to apply to the greater community of online users, the rationale being that the behavior of academics is scarcely similar to that of the general population. However, there are reasons to believe these concerns to be exaggerated. As I discussed earlier, the number of faculty working on

Internet2 is dwarfed by the numbers of staff, students, and members of their families in university housing; the universities connected to the network are not only places of work, but homes for hundreds of thousands of people. This is borne out by several observations: first, as will be discussed in Chapter 4, the proportion of Web clicks directed at adult sites is not significantly different from the population at large; and second, the measured exponents of the behavioral networks were not substantially affected by the introduction of commercial traffic between 2005 and 2008. Because of the competitive value of flow data, there is certainly no less biased population of this scale available to the network research community.

A potential source of bias in any network measurement is the vantage point of the collector. This is unlikely to be a concern in the case of the studies described in this chapter, because of Internet2's status as a major transit network. However, it does mean that the results are not necessarily indicative of what might be measured in edge networks that serve as a source or sink of data. Even though there are many Web servers hosted at Indiana University, it is unlikely that they are diverse enough to allow me to replicate these results using only data from the local network.

Another possible issue has to do with collection capacity, a concern that will come into greater prominence in Chapter 4: is the collection system capable of gathering and storing all of the network flow data, or do its limited capabilities cause it to dispose of some data in a systematic way that affects my analysis? Despite the enormous amount of data gathered in a single day, this turns out not to be a concern. The flow sequence numbers transmitted in the headers of the *netflow-v5* packets allowed me to conduct a test in which I was able to verify that no flows were lost during peak operating hours, even when the workstation gathering the data was computationally burdened.

I have already alluded to a potential source of bias that also stems from packet sampling: the fact that packets traversing multiple network segments have a greater chance of being sampled because

they touch more routers. If this multi-hop data is qualitatively different from single-hop data, this could color the results. In particular, it may increase the emphasis given to international transit data that crosses the entire United States at the expense of domestic traffic. However, international traffic is small in volume in comparison to domestic traffic, and there is simply no compelling reason to suspect that network traffic between New York and Los Angeles is dramatically different from traffic between Chicago and Cleveland. Any effect could be mitigated by using only flows on their first or last hop in the Internet2 network, but doing this properly would require SNMP access to the core routers and involve discarding a majority of the data. The main consequence is that summary statistics such as the total amount of traffic observed become somewhat suspect; whether the actual volume of traffic is 100 times greater or only 33 times greater depends on how many hops are involved in each flow. However, these values are already of anecdotal rather than practical interest.

The final concern is a basic one of trust in the data. Any of the fields in the IP, UDP, and TCP headers can be manipulated by an end system, and any application can masquerade on any port. Neither the source nor destination IP addresses of any flow necessarily correspond to either the computer that originated the traffic or any computer at all. The proportion of flow records affected by these concern is difficult to estimate. Indeed, pointing the way to improving our ability to do so is a primary contribution of the application network analysis I have just described. We do, however, have strong reason to suspect that a majority of traffic represents the data that it claims to, simply because that is the default behavior of every network operating system. Falsifying endpoints and masquerading ports requires both technical expertise and active effort. That the majority of users lack either the prowess or motivation to do so is borne out by the strength of the Bittorrent clusters in the application network: even after a decade of lawsuits by the RIAA, hundreds of thousands of users are perfectly willing to participate in the most notorious file-sharing network without

obscuring their behavior.

Effects of packet sampling

The most troubling issue of all those enumerated above is whether biases introduced by packet sampling as it is actually implemented affect the reliability of the results presented here. Of particular concern is the affect that such biases may have on measurements derived from distributions in the behavioral networks. The derivation process is fairly complex, and it is not at all straightforward to work out a theoretical model for how packet sampling in network flow records affects, for example, the distribution of clustering coefficients.

Study of the literature proved to be of little assistance in this case. There does exist a substantial body of research into how sampling can affect various graph-theoretic measures. However, it consists in the main of inquiries into the effects of sampling on structures with a known generative model, since different forms of distribution are affected in different ways. While this is unsurprising, it also offers no guidance in a situation such as the study at hand, in which there is no known model for the data. A primary contribution of my analysis is to show that the observed distributions are well-approximated by power laws over many orders of magnitude, but I cannot justify the results by assuming such a model to be ground truth.

Because of the limited support in the literature for understanding the effects of packet sampling and the inability to be certain that the sampling was taking place exactly as advertised, I concluded that an empirical study of these effects was the most appropriate way to proceed. The first step was to obtain a block of IPv4 addresses in one of the prefixes assigned to Internet2, for use as a traffic sink. Using an address in this space avoided any potential problems from asymmetric routing, since many individual institutions connected to Internet2 have a return path to Indiana University through the NLR network rather than Internet2. The basic structure for all of the experiments

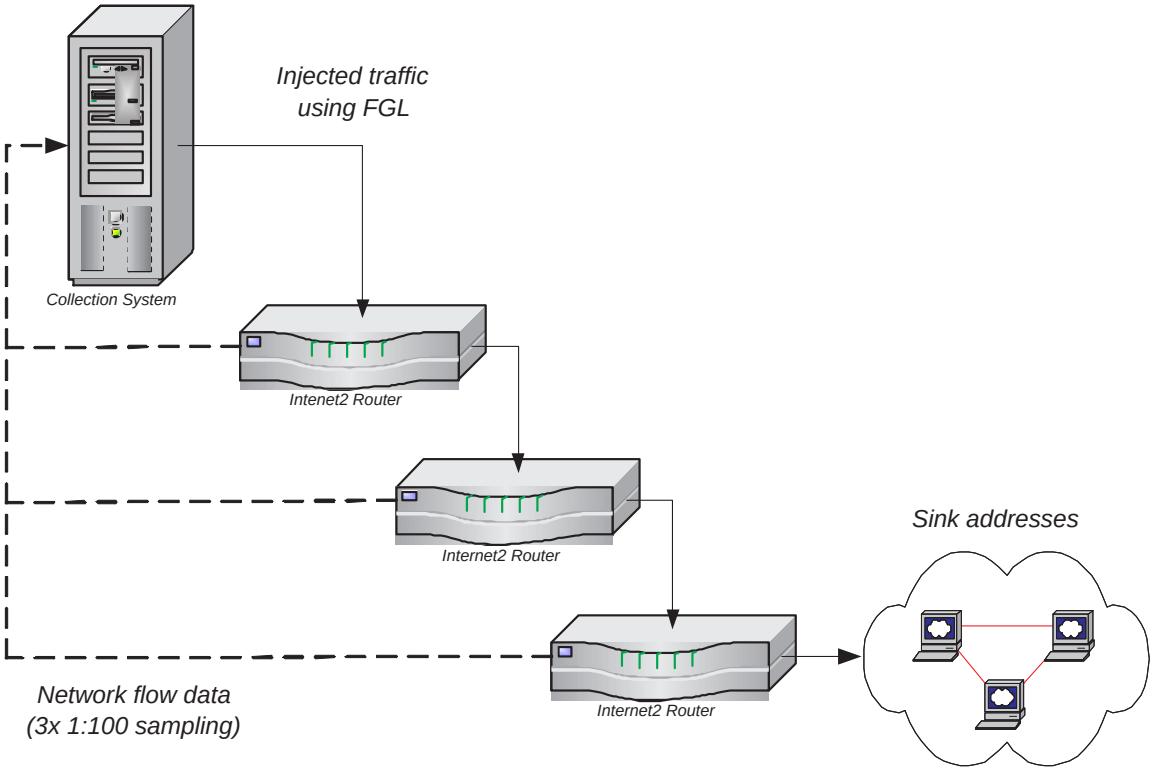


Figure 3.23: Architectural overview of the flow bias experiments. The collection system directs test traffic to sink addresses and receives network flow records from Internet2 based on a sampling of the injected data.

in the study is illustrated in Figure 3.4 and quite straightforward: while collecting the full feed of Internet2 network flow data, I send traffic with various known characteristics to the sink addresses. After the end of transmission, I look at the representation of this known traffic in the network flow records and examine the extent to which its properties have been preserved.

Conducting these experiments required access to a flexible method of generating and transmitting long-tailed distributions of network packets. Because of the large number of packets in such a distribution and the need to conduct the experiment in real time, interactive tools were unsuitable for this purpose. Other existing packet injection tools were not suited for generating packets according to a known distribution; the notion that flows exhibit a power law distribution is not

general knowledge. Injection tools are generally security tools rather than measurement instruments, and they are designed for sending particular packets rather than a whole volume of them. I also had the need for to vary fields in the packet headers so as to keep many simultaneous flows to the same address distinct from one another, which meant that any unpredictable use of ephemeral port numbers on my end was unacceptable.

The most viable solution was thus to construct my own means of generating the packets and introducing them to the network. Rather than write a new application for each experiment, I designed an scripting language for automating the process. This language, which I call Flow Generation Language (FGL), is fully described in Appendix A. The interpreter takes scripts written in FGL and using them to generate network event files that contain a sequence of packets and delays. An event-playing application can then "play" these network event files, directing the traffic to any available network interface. The language itself is designed to make conducting experiments of the kind described here as simple and succinct as possible, primarily through syntactic features that support implicit iteration on the Cartesian cross-product of any expression involving one or more sets or lists. It also supports procedures as first-class data values, which can simplify the structure of many scripts.

My first aim was to confirm that packet sampling is indeed being performed with $p = 0.01$ when generating flow records. The path through Internet2 to my target address included three core routers, which means that the chance of any individual packet being detected in this path is given by $p = 1 - (1 - 0.01)^3 \simeq 0.297$, assuming that the sampling rate is correct. That gives the chance of detection for a flow of a single packet; for a flow of n packets, the chance of detection should be $p_n = 1 - ((1 - 0.01)^3)^n = 1 - (1 - 0.01)^{3n}$. To test this, I used the FGL system to introduce flows ranging in length from 1 to 200 packets to each of the sink addresses. I then examined the network flow records to see which of these flows were represented in the data.

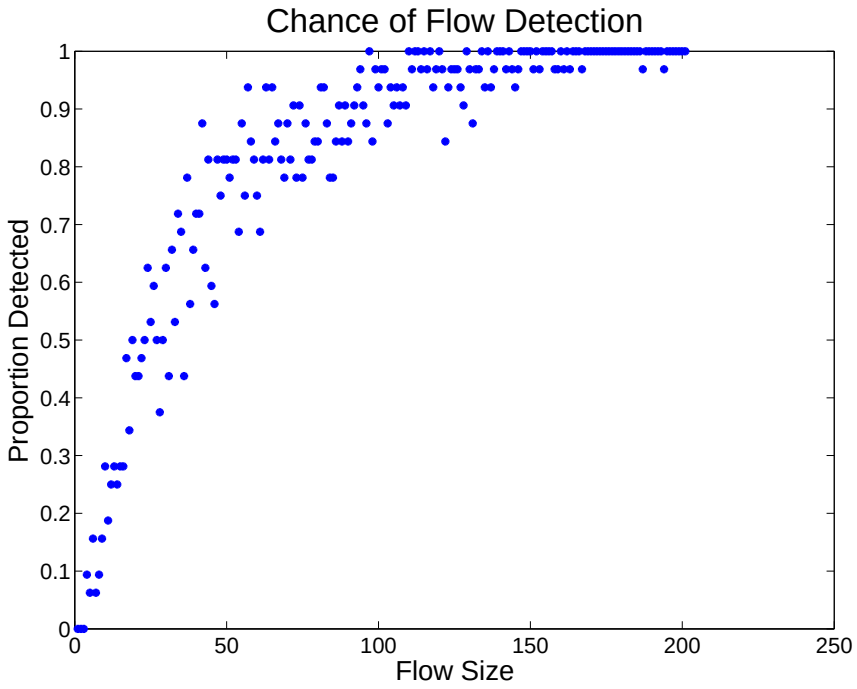


Figure 3.24: Chance of detection for flows of length varying from 1 to 200, as measured by number of packets. The x-axis indicates the flow size and the y-axis indicates the proportion of flows of this size that were reflected in the network flow records. The reference curve shows the theoretical prediction if $p = 0.01$ for each router hop in the path.

The results of this experiment are shown in Figure 3.4 and show strong agreement between the empirical and theoretical results. Although I did not test packets that themselves have unusual properties, it does seem that packet sampling occurs as advertised for normal traffic. This experiment used packets of the minimum size of 64 bytes; subsequent experimentation with differently-sized packets was unable to reveal any bias toward either large or small packets.

Having established that the mechanism of packet sampling appears to be sound, the next step was to study how sampling perturbs the measurement of the key distributions from the behavioral networks. Because the results in this chapter strongly suggest that the actual distributions of degree and strength obey power laws over several orders of magnitude, the subsequent experiments share

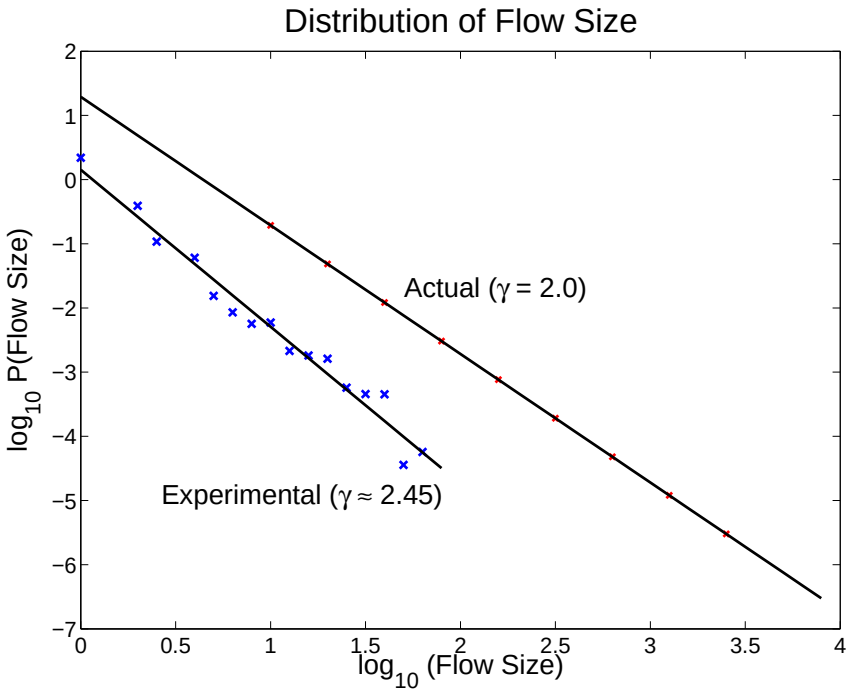


Figure 3.25: Distribution of flow size for second sampling bias experiment. The reference lines illustrate the best power-law fit for the actual distribution ($\gamma = 2.0$) and for the distribution derived from network flow data ($\hat{\gamma} = 2.45$).

a common form: introduce a collection of flows whose volumes obey a power-law distribution of known exponent, then attempt to recover that exponent from the network flow records.

In the first such experiment, I generated a network event file so that each of 10 sink hosts would receive 256 10-packet flows, 128 20-packet flows, 64 40-packet flows, and so forth. The PDF for this distribution of flow sizes thus obeys a power law $P(s) \sim s^{-\gamma}$, with $\gamma = 2.0$. I then introduced this traffic to the network and examined its reflection as recorded in the network flow records, finding the best power-law approximation to the empirical observation. The results, shown in Figure 3.4, show that packet sampling causes significant overestimation of the slope of the distribution, yielding an estimate of $\hat{\gamma} \simeq 2.45$. In this case, packet sampling causes us to believe that large flows occur less frequently than they actually do: we underestimate the variance of the distribution.

It would be premature to conclude on the basis of this single experiment that any traffic distributed with an exponent of $\gamma = 2.0$ will necessarily experience similar effects because of packet sampling. The distribution being introduced to the network can be scaled in several ways without affecting the slope of its power-law approximation. For example, we can double the number of flows in each class or double the length of each flows, and the power-law fit will remain unchanged. Either of these actions could well affect our empirical observations because of the non-linearity in the chance of detecting a flow based on its size.

In the next experiment, I increased the number of flows in each class of length by a factor of eight, so that the network event file now contained 2048 10-packet flows, 1024 20-packet flows, and so on, again to each of ten hosts. Because each flow is detected or not as an independent event, the number of flows should not affect the empirical measurement of the slope of the distribution, other than to reduce the amount of noise in the results. The power-law fit in the previous experiment had not been particularly strong, and this experiment should yield a more reliable estimate—hopefully, one that admitted $\hat{\gamma} = 2.0$ as a possibility. However, the results, shown in Figure 3.4, indicate that the network flow records still overestimate the slope of the distribution, yielding an estimate of $\hat{\gamma} \simeq 2.2$.

The persistence of this overestimate even when the number of flows was scaled up strongly implies that for this distribution of flow sizes, packet sampling will cause us to overestimate γ and thus underestimate the variance of the system. However, the flows involved in these two experiments were short enough so that the non-linearity of the chance of detecting a flow may be an important factor. There is only a 45% chance of detecting a 20-packet flow across three router hops, a probability that increases to over 99.7% if the flow is ten times longer.

For this reason, in the next experiment, I not only maintained a larger number of flows in each class, but also increased the length of each flow by a factor of ten. This time, the network event file

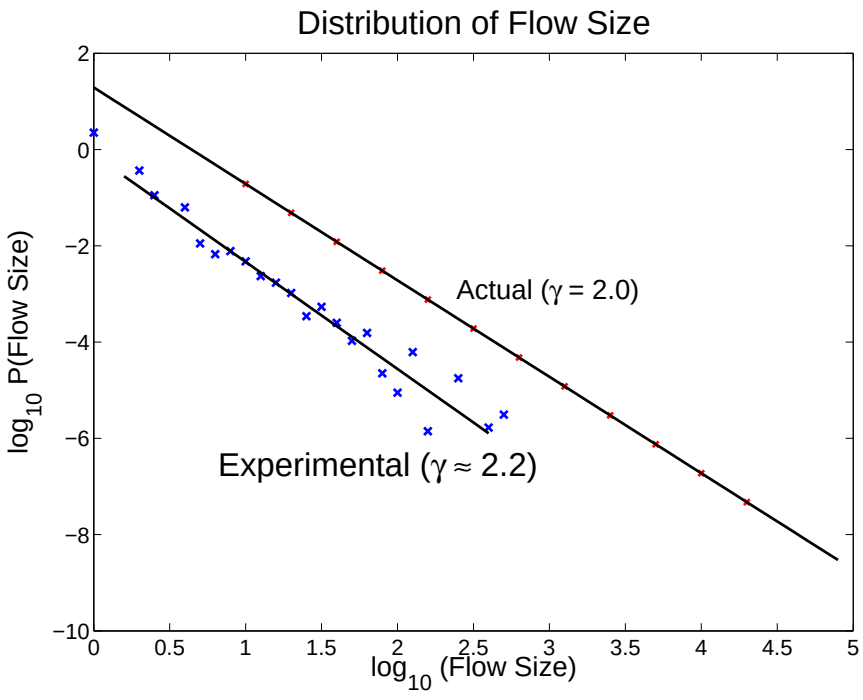


Figure 3.26: Distribution of flow size for third sampling bias experiment, with a larger quantity of flows. Once again, the reference lines illustrate the best power-law fit for the actual distribution ($\gamma = 2.0$) and for the distribution derived from network flow data ($\hat{\gamma} = 2.2$).

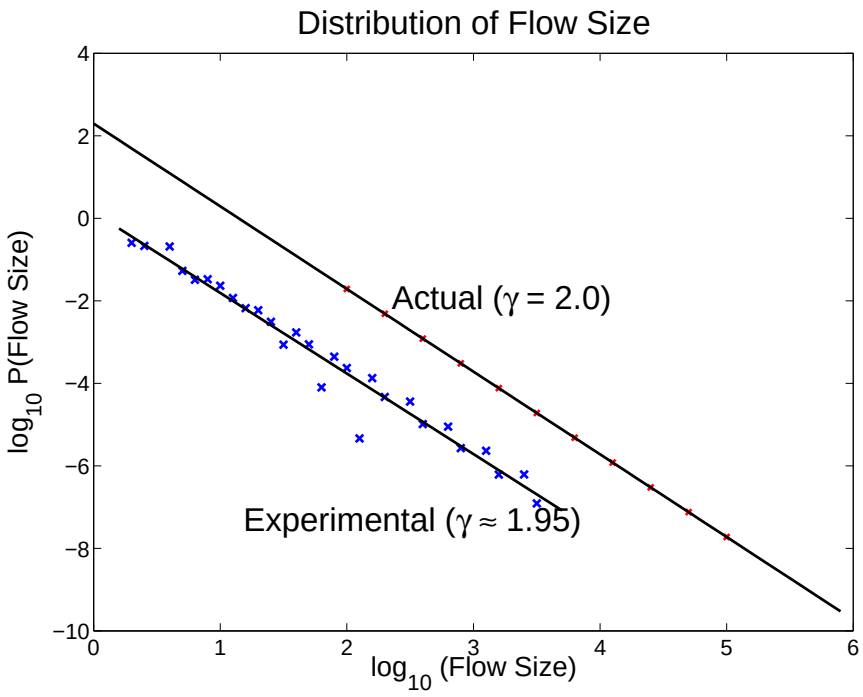


Figure 3.27: Distribution of flow size for fourth sampling bias experiment, with both a larger number of flows and longer flows. The reference lines illustrate the best power-law fit for the actual distribution ($\gamma = 2.0$) and for the distribution derived from network flow data ($\hat{\gamma} \simeq 1.95$).

contained 1024 100-packet flows, 512 200-packet flows, and so forth, again to each of ten different target addresses. The results of this experiment are shown in Figure 3.4. This time, the empirical data yield an estimate of $\hat{\gamma} \simeq 1.95$ with a fairly tight fit, suggesting that the cause of the overestimate was indeed the relatively small size of the flows in the previous experiments.

In order to confirm that the discrepancies in measuring the exponent of the distribution of flow sizes were a consequence only of the brevity of the flows rather than the particular slope, I conducted two further related experiments. In these experiments, which are reported in Figures 3.4 and 3.4, I injected power-law flow distributions with exponents of 1.57 and 2.37, respectively. In both cases, it was possible to retrieve the correct exponent from the reflected flow records within

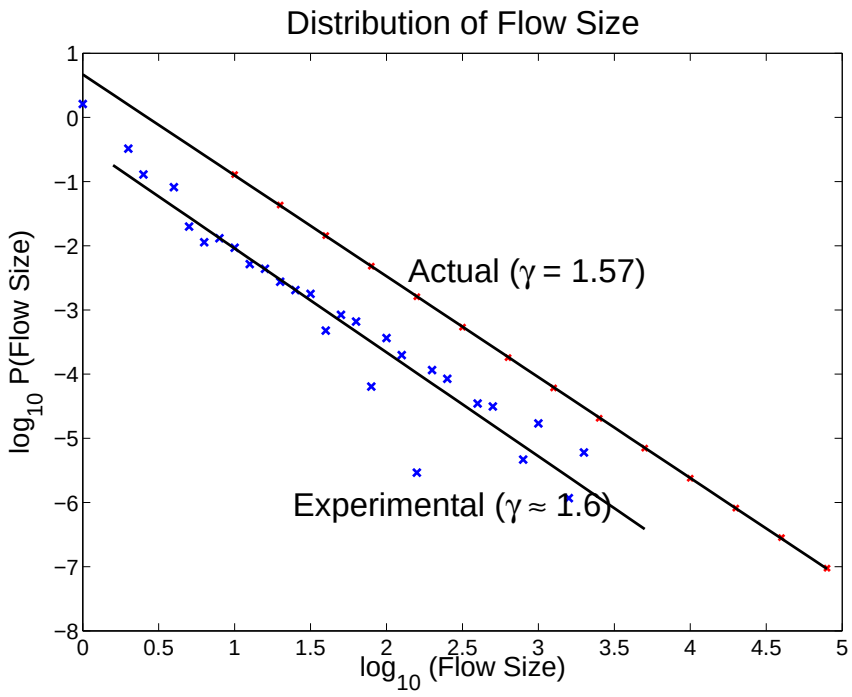


Figure 3.28: Distribution of flow size for sampling bias experiment with larger flows and a wider distribution. The reference lines illustrate the best power-law fit for the actual distribution ($\gamma = 1.57$) and for the distribution derived from network flow data ($\hat{\gamma} \simeq 1.6$).

the margin of error: $\hat{\gamma} \simeq 1.6$ in the first case, and $\hat{\gamma} \simeq 2.3$ in the second. These results lend further credence to the hypothesis that short flows lead to an underestimate of the variance of the distribution.

There are several consequences to these findings. First, the effect of the bias is to *overestimate* the slope of the power-law approximation to the actual distribution and thus *underestimate* its heterogeneity. Assuming that my analysis were vulnerable to this systematic bias caused by the interplay between packet sampling and short flows, it would mean that my conclusions are actually too conservative: the distributions would be even wider than measured, and their means even less useful. However, this bias is unlikely to have a much effect on my analysis because of the use of aggregation in calculating the behavioral networks. The weights in the behavioral networks I construct

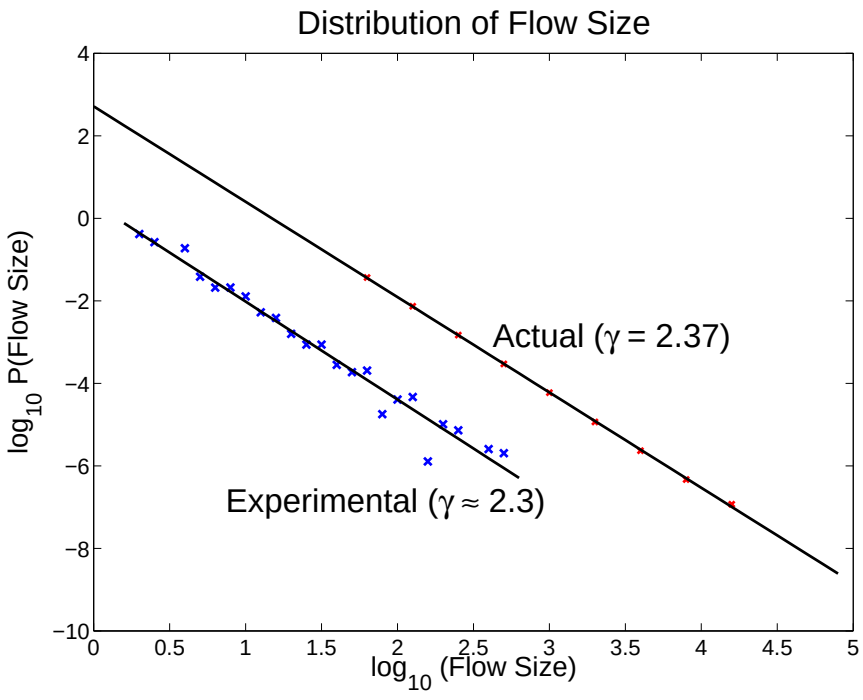


Figure 3.29: Distribution of flow size for sampling bias experiment with larger flows and a more narrow distribution. The reference lines illustrate the best power-law fit for the actual distribution ($\gamma = 2.37$) and for the distribution derived from network flow data ($\hat{\gamma} \simeq 2.3$).

only occasionally represent small, single flows; more commonly, especially in the tail of the weight distribution, they represent the summation of a number of larger flows that share common endpoints. These weights are thus much less vulnerable to this systematic bias than are individual flows.

These findings do, however, have implications for the efficacy of techniques involving the relational view of network flow data, in which each flow record is taken as a distinct entity. Short-lived flows of less than 20 packets, which have a less than even chance of detection even across three router hops, are by no means uncommon in ordinary Internet traffic. The transfer of a cascading style sheet, brief Web page, or AJAX request would certainly fall within that range. This implies that systems treating flow records as independent tuples will tend to underestimate the actual heterogeneity of network activity, possibly leading to the mistaken conclusion that measures of central tendency for network flows are more meaningful than is actually the case. This is another advantage of the graph-based approach to network flow analysis.

4

Web click data

Which application dominates the Internet in terms of traffic is an open question because of concerns about regional preferences, vantage points, masqueraded traffic, and so forth. However, there can be no question as to which application dominates the Internet in terms of public perception: as far as many millions of users are concerned, the Internet and the World Wide Web are the same thing. No other application can compare to the Web in terms of the number of person-hours spent either viewing content or preparing it for viewing by others. Even more importantly, the technologies and site designs we label as "Web 2.0" have made the Web much more of a participant sport. Instead of being a simple broadcast medium, the Web is fostering the development of a wide variety of ways for users to share information with others and interact with them online. From discussion forums to social networking sites, the Web has become the nexus of digital communication for tens of millions of people around the world.

These factors make the study of the Web in particular an essential element of any attempt to understand the structure of behavioral data from the Internet. The results in the previous chapter illustrate the presence of many interesting phenomena in Web traffic, from heavy-tailed distributions to superlinear scaling relationships. Unfortunately, we cannot dig much deeper using the

network flow data that underpins the research just discussed. This is not a simple consequence of the requirement to anonymize collected flow records: even if I were able to work with a completely uncensored feed of flow data, design features of the Internet make it impossible to push the analysis any further.

One major problem is that there is no reliable mapping between an IP address and a Web server. In the early days of the Web, this was not the case: in most cases, a host running a Web server hosted a single site, and a site resided on only one server. Both of these generalizations are now false. The introduction of virtual hosts in Version 1.0 of the HyperText Transfer Protocol (HTTP) means that a single system can host thousands of unrelated Web sites, a situation that cannot be quantified in a concrete way without access to the DNS records of the Internet as a whole. Large sites are now commonly hosted on many dozens of servers, through a variety of means ranging from redirects to round-robin DNS entries.

The greater problem lies with the design of the TCP/IP protocol stack itself. Because the formal definition of the stack ends at the transport layer, protocol details at the application layer are completely invisible to most routing hardware. Network flow records have no provision for recording any features of individual Web transactions beyond those exposed at the network and transport layers. Foremost among these features is the actual URL involved in an HTTP request, which must form the basis of any in-depth analysis of Web traffic.

Thankfully, I have been fortunate enough to gain access to a richly detailed source of Web request data that has made possible an in-depth study of Web traffic that would have been impossible using other data sources. The results of this analysis, which include not only identification of structures in user behavior on the Web but considerable success in agent-based modeling of this behavior, are presented in this chapter. These results provide insight into the mechanisms by which the varied tastes and habits of individual users can combine to create the extreme heterogeneity of

traffic described in the previous chapter and point the way to better models of user behavior that may yield improved Web applications and search engines as well as increase our understanding of the Internet's most important behavioral network.

4.1 Data source

As I have already discussed in Chapter 2, researchers have used a variety of other data sources in studies of Web traffic, ranging from server logs to direct capture of network packets. The work presented in this chapter is based entirely on the latter method, which carries some attendant advantages and disadvantages.

HTTP requests

In order to understand these advantages and disadvantages, we must first understand that HTTP is fairly unusual among application protocols in that its connections are *stateless*. While HTTP does support the use of "keep-alive" so that multiple requests can be made over a single TCP connection between a client and server, this feature is provided solely in the interest of efficiency. Each individual HTTP request is still self-contained: interpreting its meaning requires no access to previous requests, and the protocol does not enter different states in response to requests. The format of each interaction is the same: a client issues a request consisting of a protocol number, a URI, and some number of header fields providing other information about the request. The server responds with a three-digit response code, some number of header fields providing information about the response, and possibly a block of response data, such as a Web page.

This stateless design offers a strong advantage for the packet-capture approach to gathering

Web behavioral data. In a great majority of cases, we can expect the entirety of an HTTP GET request to fit inside a single TCP segment and, by extension, a single packet. The ability to establish a one-to-one association between packets and requests makes it feasible to extract orders of magnitude more data from a high-speed network connection than would be possible if HTTP were a stateful protocol requiring stream reassembly in order to interpret requests.

The potential of extracting an enormous volume of data is not the sole advantage of obtaining HTTP requests directly from the network. Having access to the HTTP headers makes it possible to extract a variety of information from each request: not only can we obtain the full URL requested, but we can also find the identity of the user agent, the referring URL, the version of the protocol used, and so forth. Of these, the referring URL (as defined by the misspelled HTTP "Referer" field) is the most interesting, as it changes the meaning of an HTTP request from being a simple request for a particular URL to being an attempt to navigate an actual link in the Web graph. This interpretation of an HTTP request as the traversal of a particular link will serve as the foundation for much of the analysis in this chapter. This is so strongly the case that I hereafter refer to HTTP requests as "clicks" to reinforce this conception of them as directed edges.

Another advantage of collecting clicks directly from the network is the opportunity to avoid many potential sources of bias inherent to other methods. I have already touched upon these sources of bias in Chapter 2, which range from emphasizing larger sites to limiting the data set to self-reported behavior. When the data are collected directly from the network, we consider all users equally, without regard to the software they have elected to install or are capable of installing. Any click traversing a network segment can be gathered without regard to either the final destination or the identity of the requester.

There are attendant drawbacks to this approach that are important to acknowledge. Chief

among them is that avoiding stream reassembly also makes it impossible to correlate HTTP requests with the server responses they elicit. In the absence of this matching, we cannot know whether a request was successful or whether the client received a redirect or server error. While the majority of requests are successful, we are forced to assume that the attempted traffic represented by redirects and server errors obeys similar distributions to that of successful navigation. This assumption seems reasonable for requests actually generated by users but is unlikely when it comes to automated probes and denial-of-service attacks. There are two potential remedies. The first is to attempt partial stream reassembly and retain only those clicks we can verify as having been successful, a task time-consuming enough to significantly reduce data-collection capacity. The other is to take into account the user agent used by each request and assume that non-representative traffic is not generated by browsers. Web clients can be made to advertise any user agent desired, so this approach is not fool-proof, but it represents a practical compromise.

Another drawback is that encrypted Web traffic transmitted using the HTTPS protocol (normally associated with TCP port 443 rather than port 80) cannot be analyzed in this fashion in the absence of complete stream reassembly and enormous computational overhead. There is no recourse in this case other than to make the assumption that encrypted Web traffic is similar in character to unencrypted traffic. This remains an open question: the research I describe here represents the most detailed analysis of the Web behavioral network available, and no similar resource exists for HTTPS traffic. There is good reason to suspect that HTTPS traffic is indeed somewhat different in character, since it usually involves authenticated access to a single site, and we can expect users to click fewer cross-domain links when navigating a secure site.

It is not the case that every click fits neatly into a single packet. On occasion, this can be caused by packet fragmentation, but the more common scenario is that a request is too large to fit inside a single packet. In many cases, this causes no problem. There is no fixed order for HTTP header

fields, but most clients tend to specify the most relevant ones (user agent and referrer in particular) first. In other cases, the URL being requested is too long to fit inside a single 1,500 byte Ethernet frame. This is particularly true of some Web services that use HTTP GET rather than POST and encode a large number of variables directly into URLs. In this case, we have no recourse other than to discard the click entirely. The frequency with which this occurs can be quantified; I do so in the next subsection.

A final and significant disadvantage is the potential damage to user privacy entailed by any inspection of network packets. All of the data collection described in this chapter has been conducted using procedures and practices meant to safeguard the privacy of individual users to the greatest extent possible while still gathering meaningful data. Even so, the potential for abuse with access to uncensored feeds of aggregate data is large enough so that a responsible researcher must be extremely circumspect about designing collection infrastructure and sharing access to both this infrastructure and the data itself. I have made every effort to do so; my fervent hope is that future researchers in the area do so as well.

Data collection

I have two primary sources of click data for the research described in this chapter, both of which involve the Indiana University network. The first of these data sources involves the entire university, is by far the larger of the two, and is available on an ongoing basis. The second is a more limited set of data gathered from one of the Bloomington dormitories. In this section, I discuss only the first of these data sets, deferring a detailed explanation of the second until Section 4.3. The technical details surrounding the collection of both sets are quite similar.

The university-wide data source derives from a click collection system located on the Indianapolis campus. This system receives, in theory, a copy of all traffic passing between the eight campuses

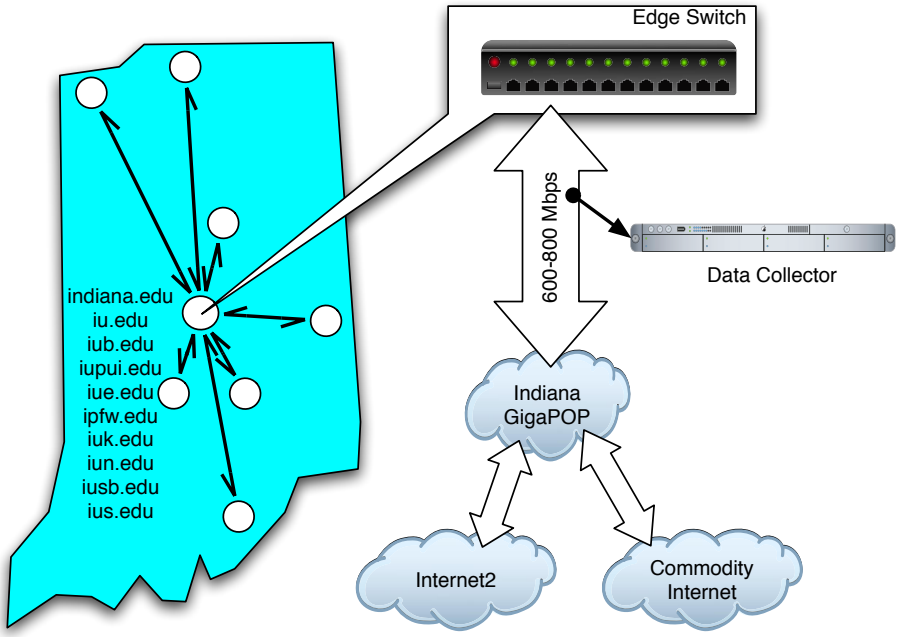


Figure 4.1: Sketch of Indiana University's Internet connectivity and the data collection framework for university-wide click data.

of Indiana University and the rest of the Internet. There is reason for this caveat: this system is a dedicated FreeBSD server with a 1 Gbps Ethernet interface specially calibrated for maximum performance in packet capture, but the aggregate traffic between the university and the rest of the Internet often exceeds the capacity of a single 1-Gbps link. Under normal conditions, the collection system's network interface is capable of processing around 800 Mbps of network data that represents the aggregate network activity of over 100,000 users. A basic schematic of the data collection framework is shown in Figure 4.1.

While some data is lost because of the limited capacity of the network interface, this excess traffic is not discarded in any systematic way. However, university traffic does obey a strong diurnal cycle, peaking in the early afternoon and reaching its minimum level in the early morning hours. Because all analysis is performed on a best-effort basis, this does mean that data gathered during

these hours of light traffic is more strongly represented in the resulting data set. Simply put, there is a better chance of capturing a 3am click than a 3pm click.

To obtain information on individual clicks passing over the network interface, I use a Berkeley Packet Filter (BPF) to capture only packets destined for TCP port 80. This was a primary consideration for using FreeBSD for the collection platform rather than Linux or Solaris: it implements BPF in kernel address space rather than user address space, eliminating the considerable overhead of copying every packet between different blocks of memory. As mentioned before, I make no attempt to capture or analyze any encrypted traffic using TCP port 443, nor do I attempt to gather any Web-related traffic on other TCP ports.

Once the collection system receives a packet destined for port 80, I anonymize it pursuant to the conditions set by UITS when I arranged for data collection. This involves removing all identifying information about the client from the IP and TCP headers, making it impossible to associate the payload data with any particular client system. Note that this is more stringent anonymization that was the case for network flow data, since I maintain no distinction at all between requests gathered from different clients, not even a temporary index or one-way hash. In theory, every click in the data set could be part of the activity of one extraordinarily busy user, though this is of course far from the truth.

The collection system then applies a regular expression search against the packet payload to determine whether it contains an HTTP GET request. If a request is found, it analyzes the packet further to determine the identity of the virtual host contacted, the path requested, the referring URL, the advertised identity of the user agent, and whether the request is inbound to or outbound from the university. The relevant bits of information are indicated graphically in Figure 4.1. The collector then writes a record to disk containing a timestamp, the requested and referring URLs, a direction flag, a flag indicating whether the user agent matches a mainstream browser (Internet

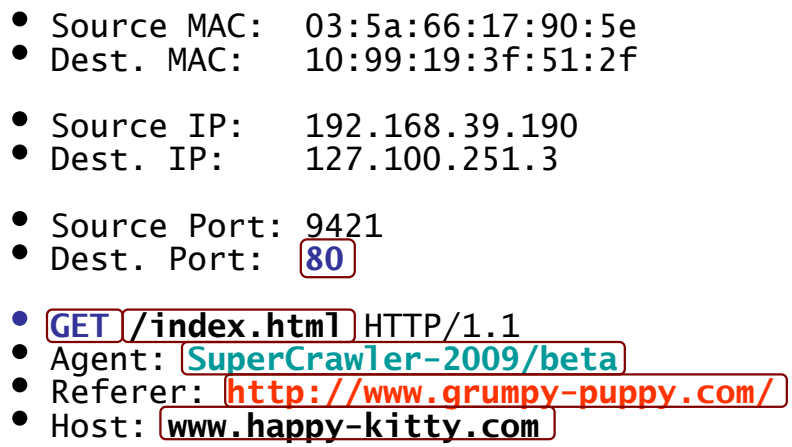


Figure 4.2: Summary of the information gathered from a single HTTP request. This visualization is conceptual and does not represent the actual packet layout.

Explorer, Mozilla/Firefox, Safari, or Opera). Note that the user agent field must be reduced in this way to save disk space: most agent strings are fairly long and include detailed revision and platform information. It is common for the collection system to observe well over 10,000 unique agent strings over the course of a single day.

All of this on-the-fly analysis is accomplished using precompiled regular expressions and well-optimized code. Coupled with the carefully configured network sack, this allows me to record roughly 30% of all clicks during peak traffic hours. On a typical weekday, this amounts to around 60 million HTTP requests, the raw records for which require about from 6–10 GB of storage depending on whether the collection is configured to store full URLs or simply hostnames.

Generation of host graph

As mentioned before, the presence of the referring URL in HTTP requests makes it possible to think of a request as not just a simple resource fetch, but the attempted traversal of an actual link of the Web graph. In other words, the observation of a request for URL A with referring URL B suggests that an actual hyperlink may exist between these URLs and that a real user wishes to navigate this link. Every click is evidence of such a link and can be seen as a directed edge in the Web graph. If we aggregate these edges into a graph structure over an extended period of time, the result should be a subset of the real Web graph with links weighted according to actual traffic generated by users.

This is a tremendous advantage over studies based on crawling the Web, which can uncover many billions of pages but can derive only an unweighted version of the Web graph. A crawler has no way of knowing whether it is the first agent ever to follow a link—and if it is, we can rightly question whether that link is actually a part of the Web in any meaningful sense. A weighted Web graph derived from user data represents the part of the Web that is actually being used, which has the potential to yield insights not visible through examination of the static Web graph.

This discussion has been in the abstract; actually deriving a graph structure from the raw click data involves a number of compromises. First, although it is possible in principle to capture the entire URLs of the referring and requested pages for each click, in first part of this chapter I confine myself to the host graph, in which only the hostnames of each URL are retained. I have done so for several reasons. First, the structure of the host graph has been considered in the literature and found to be broadly comparable to the page graph. Second, the host graph is far more tractable to analyze, given the storage and computing resources available. It also cannot help but to allow more aggregation of weights among the edges during a shorter period of time. Finally, as we shall see, the host graph is sufficient to reveal a number of interesting insights into Web traffic.

To derive a weighted version of the Web host graph from the raw click data, I first reduce the data into "click lists" containing only the indices of the referring and target servers for each observed HTTP request. These indices are pointers into an external database that contains the fully qualified domain names of the Web servers involved. Two different click lists are generated from the raw data files. The first one, FULL, includes every HTTP request detected on port 80, regardless of directionality, user agent, or the URL requested. While this represents all of the raw data, it would be a poor basis for understanding the host graph: it is biased toward Web sites hosted at Indiana University, it includes crawler traffic, and it includes requests for all types of resources, not just Web pages. The second click list, HUMAN, is a subset that includes only those requests that are made by a common browser, generated by a user inside the university, and for URLs that are likely to be actual Web pages rather than media files, style sheets, and so forth. Because the server responses are unavailable, this final criterion must be accomplished through a heuristic based on the extension of the URL requested. While this heuristic is necessarily imprecise—for example, not every *.php* URL is actually a Web page—it seems to work well in practice.

The click lists use the zero index to refer to an illusory Web server that represents the "empty referrer." This is the source host for every HTTP request that does not include referrer information. These requests are the result of jumps rather than navigation; their sources include bookmarks, browser start pages, mail systems, office applications, clients with privacy extensions, and so forth. The distribution of traffic for this empty-referrer host will become especially important when we consider modeling applications.

As mentioned before, some requests turn out to incomplete or malformed, either because they were too large to fit in a single packet, or because they were generated by old or primitive client software that issues HTTP/0.9 requests that do not include a virtual host field. These broken requests make up a small proportion of the total: 1.74% of the FULL click list and only 0.24% of the

HUMAN click list.

The click lists represent series of directed edges in the actual host graph for the Web. When we merge a set of these edges, we obtain a subset of the graph weighted according to observed user traffic over some interval of time. We are thus able to apply various levels of aggregation to the click lists to generate hourly, daily, monthly, and cumulative versions of the observed host graph. These graphs are stored as sparse connectivity matrices for analysis using MATLAB and the suite of tools for manipulating large graphs described in Appendix A.

4.2 Analysis of the host graph

My analysis of the host graph is three-pronged. First, I consider the structural properties of the graph, operating along much the same lines as I did for the behavioral networks in Chapter 3. Next, I consider behavioral properties revealed by the weighted graph, which include a consideration of the proportion of traffic derived from different classes of server and temporal properties of the traffic. Finally, I consider the implications of this weighted host network for the assumptions implicit in the random surfer model, which paves the way for the applications described in Sections 4.3 and 4.4.

Structural properties

The click data involved in the analysis of the host graph presented here was collected over a period of about seven months, from September 26, 2006, to May 19, 2007. Due to technical complications, no data were collected for two intermediate periods between January 15, 2007, and January 28, 2007; and April 1, 2007, to April 8, 2007. This time period serves as a representative slice of the available data; subsequent analysis has shown that the properties discussed here are quite stable,

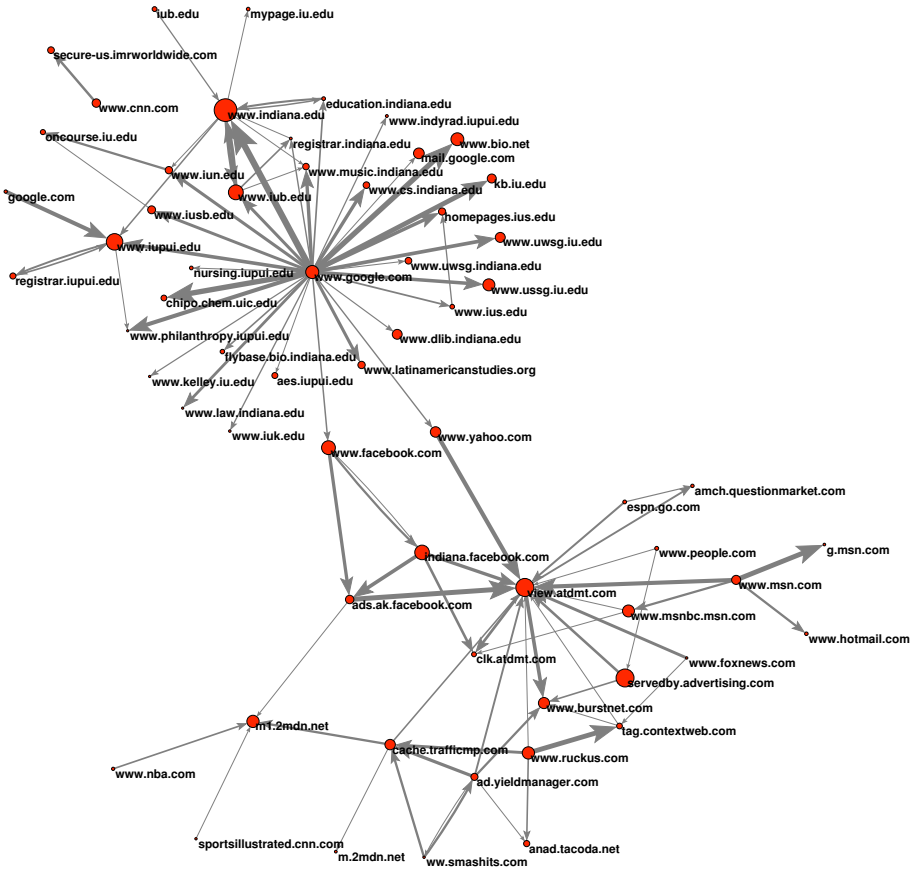


Figure 4.3: Visualization of the core of the traffic-weighted Web host graph, showing the most requested hosts and the most clicked paths between them. The node size is proportional to the log of the traffic to each site, and the thickness of the edges is proportion to the log of the number of clicks on the links between the sites.

with a gradual increase in overall traffic as time goes on. A flavor of the dominant links in the host graph can be obtained from the visualization in Figure 4.2, which depicts a greatly reduced graph showing only the most popular destination sites and the most clicked paths between them.

The basic dimensions of both the FULL and HUMAN versions of the data set, and the associated versions of the host graph, are summarized in Table 4.1. Both versions of the data set are quite large: there are roughly 12.8 billion clicks, 7.5 million nodes, and 37 million edges in the FULL data set; and 910 million clicks, 4.0 million nodes, and 11 million edges in the HUMAN data set.

Table 4.1: Summary statistics of the FULL and HUMAN host graphs.

		FULL		HUMAN	
		Number	Percent	Number	Percent
requests	with empty referrer	2,632,399,381	20.4%	490,290,850	54.0%
	to unknown destination	232,147,862	1.8%	2,078,725	0.2%
	total	12,884,043,440		907,196,059	
hosts	referring	5,151,634	67.8%	2,199,307	54.5%
	destination	7,026,699	92.5%	3,743,074	92.8%
	total	7,595,907		4,031,842	
edges		37,537,685		10,790,759	

The dramatic reduction in size between the FULL and HUMAN sets is almost entirely due to the removal of requests for Web-linked resources other than pages. Every request for an actual page by a human operating a browser generates an average of 14.2 HTTP requests for embedded media files, style sheets, script files, icons, and so forth.

One notable observation at this level is that a majority of human-generated clicks do not have a referrer page, indicating that users typed the URL directly, clicked on a bookmark, opened the URL from another application, or simply use that URL as their browser's start page. In Section 4.3, I explore the effects of treating these empty-referrer clicks as the beginnings of individual browsing sessions.

The first structural question about the host graph reconstructed from this sample of traffic is whether we can recover the well-known topological features of the link graphs built from previous large-scale crawling projects [9, 30, 51, 134]. The most stable signature of the Web graph has been its scale-free in-degree distribution, which many studies consistently report as being well-fitted by a power law $\Pr(k_{in}) \sim k_{in}^{\gamma}$ with exponent $\gamma \approx 2.1$. The in- and out-degree distributions for the FULL and HUMAN versions of the host graph are shown in Figure 4.2. As shown in this figure, we do recover this behavior from the FULL host graph ($\gamma = 2.2 \pm 0.1$); although Web traffic may not follow every link that exists, it still produces a picture of the Web that is topologically consistent

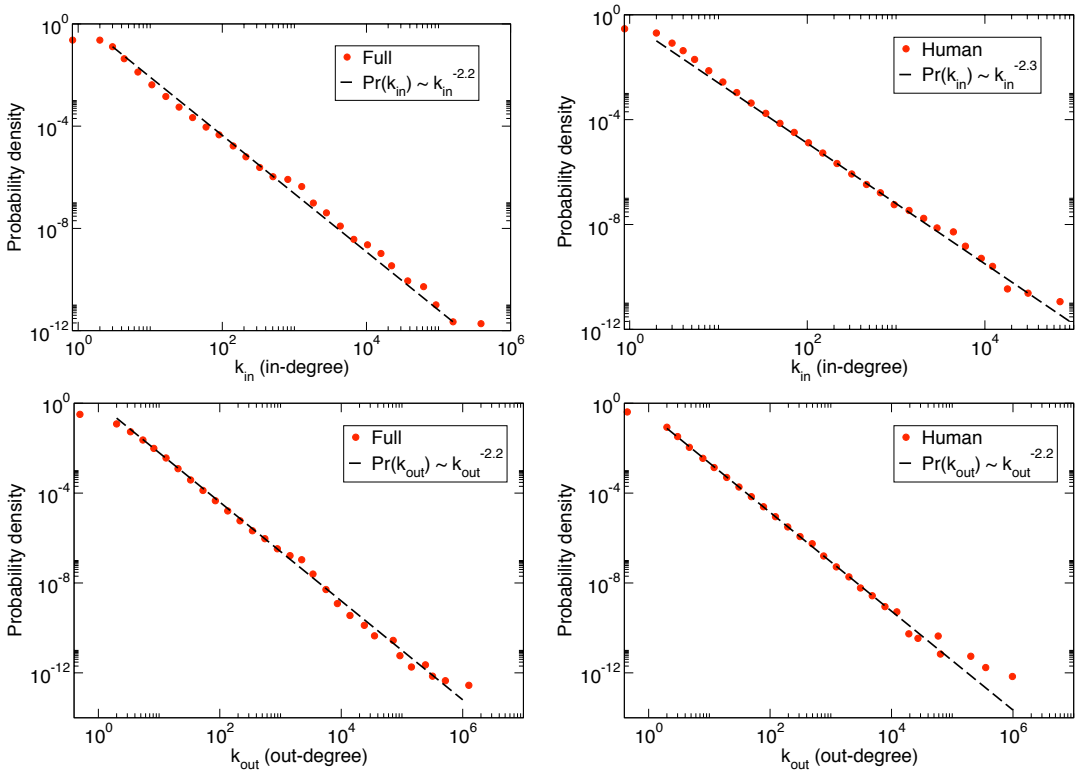


Figure 4.4: Distributions of in-degree k_{in} (top) and out-degree k_{out} (bottom) for the FULL (left) and HUMAN (right) host graphs. The best-fit power law approximations for the distributions are included as a visual reference.

with those obtained from large-scale crawls.

The in-degree distribution in the HUMAN host graph represents a more complex situation: in this case, the best-fit power law approximation has a slightly larger exponent $\gamma = 2.3 \pm 0.1$. This discrepancy hints at an important caveat of the traffic-weighted network. While the structure of the networks induced by traffic and crawler activity may be similar, they are based on very different sampling procedures, each with their own attendant biases. One cannot compare the two networks directly on a node-by-node basis. In the crawler's view of the Web, a link must simply exist in order to be part of the graph; in the traffic-based view, a human being at Indiana University must have cared enough about that link to actually click on it.

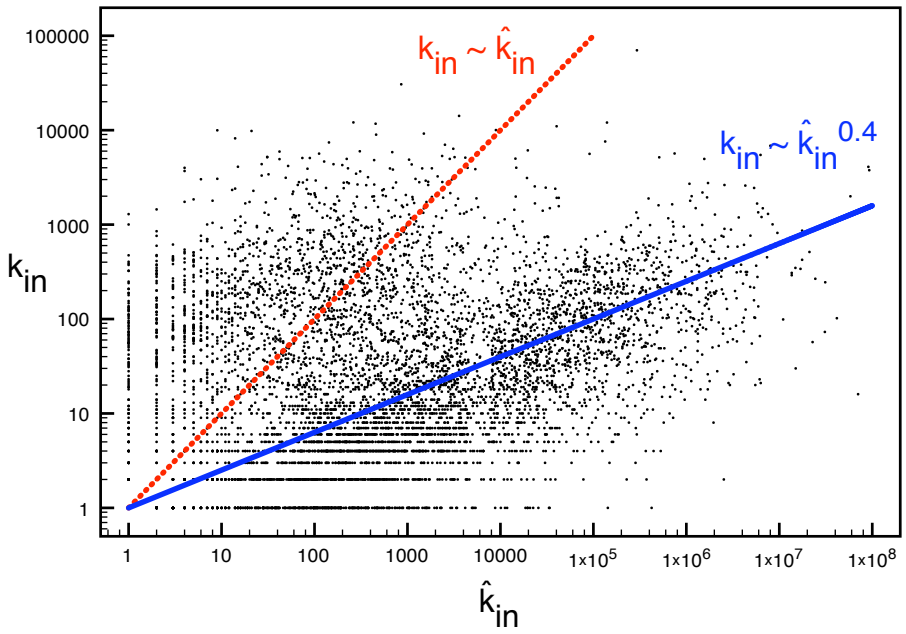


Figure 4.5: Scatterplot of k_{in} values estimated from the HUMAN host graph versus $\hat{k}_{in}$ values obtained through the Yahoo! API. Lines indicating proportionality and the best-fit power-law scaling are shown as a reference, with the caveat that such a model may not be the best description of the relationship.

To illustrate this point, we can sample nodes from the HUMAN graph and compare their in-degree with that given by a search engine via the Yahoo! API. The results of doing so are shown in the scatter plot in Figure 4.2, which reveals only weak correlation (Pearson's $\rho = 0.26$ on the log values); we cannot assume proportionality. If we conjecture a power-law scaling relationship $k_{in} \sim \hat{k}_{in}^{\eta}$, where $\hat{k}_{in}$ is the in-degree obtained from crawl data, we can see that a sublinear bias $\eta < 1$ fits the data better than simple proportionality $\eta = 1$. While we cannot conclude that such a power-law scaling is the most appropriate model for the relationship between k_{in} and $\hat{k}_{in}$, this exercise does highlight a sample bias whereby the in-degree of popular nodes is underestimated by a greater amount than that of low-degree nodes.

The lack of proportionality explains the higher exponent in the power-law distribution of in-degree. If we assume again that k_{in} and $\hat{k}_{in}$ are indeed related by the power-law scaling conjectured above, it follows immediately that $\Pr(k_{in})dk_{in} = \Pr(\hat{k}_{in})d\hat{k}_{in}$. Therefore

$$\begin{aligned}\Pr(k_{in})dk_{in} &\sim k_{in}^{-\gamma}dk_{in} \sim \hat{k}_{in}^{-\eta\gamma}d(\hat{k}_{in}^{\eta}) \\ &\sim \hat{k}_{in}^{-\eta\gamma+\eta-1}d\hat{k}_{in} \sim \hat{k}_{in}^{-\hat{\gamma}}d\hat{k}_{in}\end{aligned}$$

and thus the k_{in} exponent changes to $\gamma = (\hat{\gamma} - 1)/\eta + 1 > \hat{\gamma}$ if $\eta < 1$.

To conclude this point, it is also worth repeating the philosophical point that this discrepancy is not a fault of the traffic data but a by-product of the implicit view that the only *real* links are the ones that people use. Which view is closer to ground truth is more a function of attitude than of methodology.

For various reasons, the literature is less consistent about the characterization of the out-degree distribution of the Web. As shown in Figure 4.2, the present data are consistent with a power law distribution $\Pr(k_{out}) \sim k_{in}^{-\gamma}$ with exponent $\gamma \approx 2.2$.

Having considered the basic structure of the host graph, we can turn to the major characteristic that distinguishes this representation from the version of the host graph we obtain from crawling: the weights of the edges. We examine first the distributions of in-strength and out-strength for the nodes in the host graph. Note that $s_{in}(i)$ represents the total number of times that a user has visited site i, making it the concrete representation of what I informally refer to as "traffic." Figure 4.2 plots the distributions of strength for the host graph. In this case, not all the curves are best fit by power-law approximations; nevertheless, all of the distributions are extremely broad, spanning a full eight orders of magnitude. The portions of the distributions best fit by power laws $\Pr(s) \sim s^{-\gamma}$ yield γ values between 1.7 and 1.8. Recall from Chapter 3 that these exponents $\gamma < 2$ imply that the

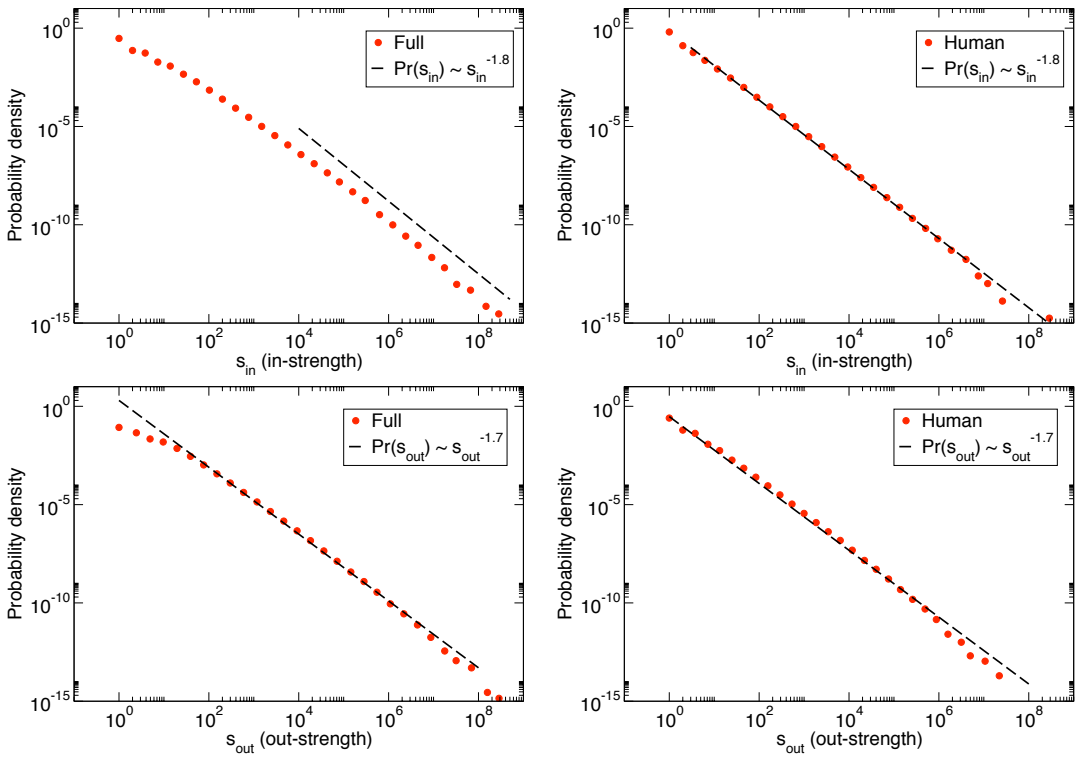


Figure 4.6: Distributions of in-strength s_{in} (top) and out-strength s_{out} (bottom) for the FULL (left) and HUMAN (right) host graphs. The best-fit power law approximations for the distributions are included as a visual reference.

mean strengths diverge as the networks grow and are bounded only by the finite size of the data collected. Such broad distributions of traffic suggest that the static link graph can capture only a small portion of the actual heterogeneity of popularity among Web sites.

It is also instructive to consider the distribution of the weights w_{ij} (*link traffic*) across all of the directed edges $i \to j$ in the host graph. As shown in Figure 4.2, these too are broad distributions over many orders of magnitude and can be fit to power laws $\Pr(w) \sim w^{-\gamma}$ with exponents γ between 1.6 and 1.9. Such extreme heterogeneity indicates that not all links are created equal: a few strong links carry a wildly disproportionate amount of traffic while most carry little at all. We cannot assume that this is a distinct phenomenon from the scale-free distribution of traffic for the

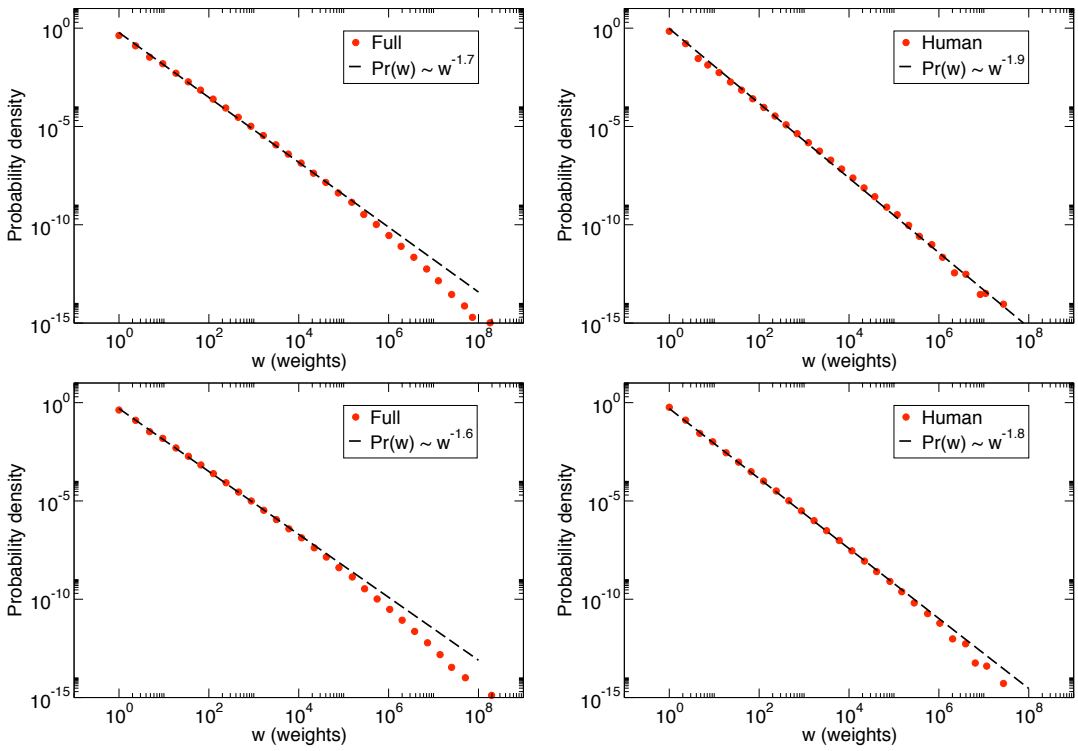


Figure 4.7: Distributions of weights excluding (top) and including (bottom) requests from the empty referrer for the FULL (left) and HUMAN (right) host graphs. Requests with non-empty referrer correspond to clicks from one page to another, whereas an empty referrer click may originate from a bookmark or directly typed URL.

originating hosts of the edges; it could simply be a trivial correlation. The local heterogeneity of traffic across links from individual hosts is discussed later in this section.

The assortativity of the Web host graph is depicted in Figure 4.2, which examines assortativity in terms of both degree (the number of links connecting two sites) and strength (the amount of traffic connecting the sites). We can see that the host graph is disassortative with respect to degree, consistent with the topology of a technological network [120]. However, this effect is much weaker in the case of strength, implying that the superimposition of human behavior on the network gives it a much more assortative flavor.

The size of the HUMAN data set made it tractible to compute the principal eigenvectors of

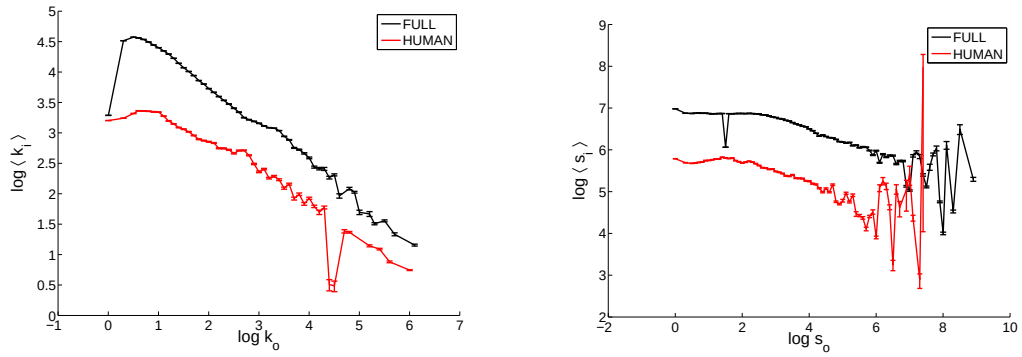


Figure 4.8: Assortativity in the Web host graph, examined in terms of degree or link count (left) and strength or aggregate traffic (right). Each plot shows the in-degree [in-strength] of nodes as a function of the out-degree [out-strength] of the nodes linking to them.

the weighted connectivity matrix. In doing so, my hope was that examining two-dimensional projections of the hosts into planes defined by pairs of eigenvectors would yield insights suitable for identifying anomalous hosts or building clusters of related sites.

Unfortunately, spectral analysis has not yielded any useful results. The entries of the top fifteen eigenvectors are broadly distributed in the interval $[-1, 1]$, roughly centered on zero, with exponents ranging smoothly from 10^{-1} to 10^{-23}. The preponderance of negative values makes visualization of pairs of vectors awkward, since the broad distributions require double logarithmic axes. My attempts at constructing an appropriate visualization show that all pairs of principal eigenvectors are strongly correlated. They reveal a variety of intriguing structures, but these take the form of extensions from the central diagonal rather than identifiable clusters, and I have been unable to identify groups of sites corresponding to these features; failed hypotheses include search engines, social networking sites, and online mail services. Unsurprisingly, an attempt to use the principal eigenvectors to form clusters using the k-means algorithm yielded inconclusive results; while the resulting clusters had some internal consistency, there were no obviously appropriate labels. I am still intrigued by the potential of spectral analysis in analyzing large-scale behavioral

Table 4.2: Proportion of requests by category of referring host. The percentages of edges are shown under k_{out} (total out-degree), and the percentages of traffic are shown under s_{out} (total out-strength). For requests with an empty referrer, the percentage of edges is computed by considering those requests as originating from the special "empty referrer" host.

	FULL		HUMAN	
Source	k_{out}	s_{out}	k_{out}	s_{out}
Empty	10.2%	20.4%	**14.5%**	**54.0%**
Search	8.2%	2.9%	**21.2%**	**4.9%**
Webmail	3.1%	2.0%	1.6%	0.6%
Other	78.5%	74.6%	62.8%	40.4%

networks, but have no further insights at this time.

Behavioral properties

The properties considered so far are analogous to those investigated in Chapter 3 for the Web behavioral network derived from network flow data, and it may be unsurprising that the results are similar: extreme heterogeneity over many orders of magnitude and the lack of meaningful central tendencies. However, the analysis does not have to end there: the URL information and large time interval for the data set make it possible to investigate a number of other properties related to user behavior.

Examination of the hostnames involved in each click allows us to explore some basic questions about how users navigate the Web. First, to what extent do people wander from page to page (i.e., surfing by following links) versus visiting pages directly (i.e, teleporting using bookmarks, typing URLs, and other means)? Table 4.2 shows the percentages of requests originating from different types of source as determined by applying simple heuristics to the domain name of the referring host.

Let us focus on the HUMAN version of the host graph. As noted in the previous subsection, the

majority (54%) of requests have an empty referrer, corresponding to pages visited directly, without clicking on a link. Such a high number suggests that traditional browser bookmarks may still be widely used, in spite of the growing popularity of online bookmark manager. On the other hand, all of this traffic corresponds to only 14.5% of the edges. The small k_{out}/s_{out} ratio indicates that each additional empty-referrer request is increasingly less likely to lead to a destination host that has not been seen before. This is reasonable behavior for sites that are bookmarked or have easily remembered URLs.

The magnitude of data available makes it possible to examine the extent to which these proportions of traffic vary as a function of time. In Figure 4.2, we can see that while the overall volume of requests in each category is affected by both the academic calendar and the time of day, the relative ratios of clicks from different sources are fairly stable.

We can also see from Table 4.2 that less than 5% of total traffic originates from search hosts. The basis for defining this category was to match the domain names of referring hosts against a list of common search engines, including Google, Yahoo!, MSN, Altavista, and Ask. Such a low percentage is a bit of a surprise, considering the wide impact generally attributed to search engines in steering Web traffic. It must be noted that the measure presented here gives only a lower bound on the influence of search engines, since it considers only requests that are *directly* generated by search; successive clicks appear as regular navigation, even if they occur along a path initiated by a search. This cannot be avoided: recall that all identifying client information is discarded as part of the data collection process, making it impossible to recover chains of clicks. Nevertheless, one might expect a much higher percentage of clicks originating from search engines.

Another notable feature of search traffic is the much higher fraction (21.2%) of edges corresponding to these clicks. The high k_{out}/s_{out} ratio suggests that each search click in more likely than others to lead to a new host. In other words, search engines appear to promote rather than

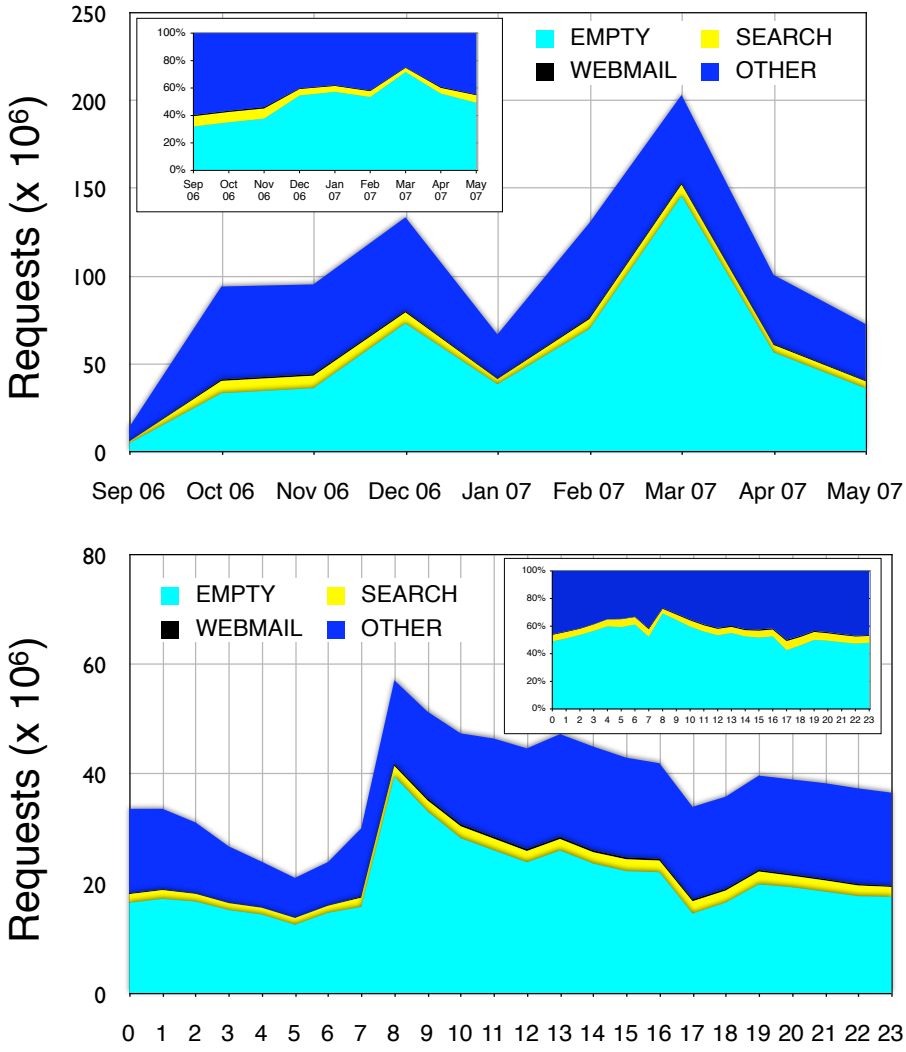


Figure 4.9: Volume of requests from various sources in the HUMAN host graph. The top figure depicts seasonal variations, aggregated by month. The bottom figure show daily variations, aggregated by hour. The insets plot the percentages of requests from each category of referrer (total strength). The monthly and hourly ratios of total degree by source (not shown) are even more stable.

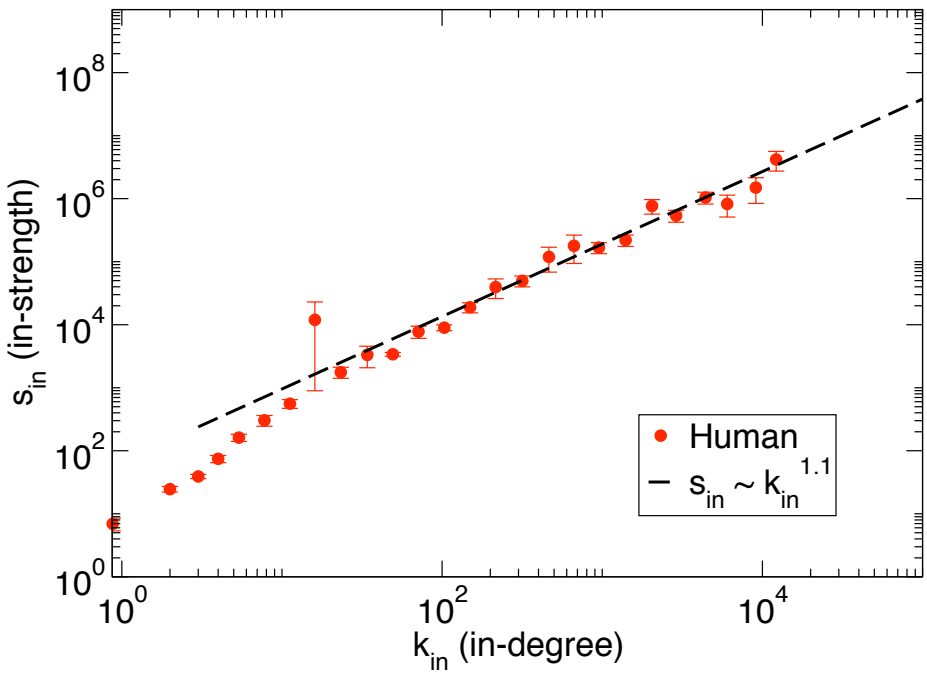


Figure 4.10: Scaling of traffic with in-degree k_{in} in the HUMAN host graph. Error bars correspond to ± 1 standard error on the log-bin average.

inhibit the exploration of unvisited sites, an egalitarian effect that is in agreement with earlier findings [130, 65].

Another method of using this data to measure the impact of search on Web navigation is to examine how traffic scales with in-degree. Earlier literature conjectured that the use of PageRank by search engine would lead to a feedback cycle in which established sites would attract an ever-growing fraction of traffic, while less popular and newer sites would be unable to improve their rankings [14, 85, 118, 41]. Such a cycle would lead to superlinear scaling of search traffic with respect to in-degree:

$$s_{in} \sim k_{in}^{\beta}$$

with $\beta > 1$. In contrast, random walk alone would yield linear scaling ($\beta = 1$).

Figure 4.2 depicts the relationship between in-strength and in-degree, allowing us to observe a trend that is slightly above, but not statistically different from, the linear model of a random walk ($\beta \gtrsim 1$). This is consistent with the observed distributions of s_{in} and k_{in}. Since both are approximated by power laws with exponents γ_s and γ_k respectively, if the two scale together with a power relationship, we then have $\Pr(s)ds = \Pr(k_{in})dk_{in}$. This leads to

$$\begin{aligned} s_{in}^{-\gamma_s} ds &\sim (k_{in}^{\beta})^{-\gamma_s} d(k_{in}^{\beta}) \sim k_{in}^{-\beta\gamma_s+\beta-1} dk_{in} \sim k_{in}^{-\gamma_k} \\ \beta &\approx (\gamma_k - 1)/(\gamma_s - 1). \end{aligned}$$

In light of the error bounds on the fitted parameter β, the empirical values of β, γ_s, and γ_k are mutually consistent. However, the finding $\beta \gtrsim 1$ appears to conflict with results from a collaborator's previous work based on traffic data from Alexa. In that case, a sublinear scaling fitted the data better, suggesting that search engines can actually mitigate the "rich-get-richer" dynamic of the Web [65]. A likely resolution to this conflict rests in the sampling bias for in-degree discussed earlier, which affects the relationship between traffic and in-degree. Recall the conjecture that we observe power-law scaling $k_{in} \sim \hat{k}_{in}^{\eta}$. We would then find $s_{in} \sim k_{in}^{\beta} \sim \hat{k}_{in}^{\eta\beta}$. If $\eta < 1/\beta$, as suggested by Figure 4.2, then $\eta\beta < 1$. In other words, we recover a sublinear scaling of traffic with respect to the crawl-based in-degree $\hat{k}_{in}$, making the traffic data consistent with prior empirical findings. Unfortunately, it is difficult to infer much more about the impact of search engines on this trend because of the small percentage of the traffic in the data set actually deriving from search engines.

The timestamp information associated with the Web requests allows us to do more than just see how the proportion of different categories of the traffic changes over time. We can compare entire host graphs constructed from different time segments of the data and thereby study the predictability of traffic over time. Using the weighted host graph to predict future traffic has potential

application for Web cache refreshing algorithms, capacity allocation, and site design. For example, if an ISP knows that a certain news page is regularly accessed every morning at 10am, it can preload it into a proxy server. Likewise, knowledge about regular spikes in traffic can guide provisioning decisions. Finally, site owners and hosting providers can adapt sites so that content can be made more easily accessible based on predicted demand at different times. These potential benefits become even more relevant as many enterprises move toward virtualized hosts that can be dynamically re-provisioned in response to demand.

As an initial attempt to quantify the predictability of Web request patterns, let us adapt the simple precision and recall measures, which are well-established in information retrieval and machine learning. The goal will be simple: to predict the structure and weights of the host graph for time interval t using a snapshot of the host graph at time interval $t - \delta$, as a function of the prediction delay δ. In other words, we are simply investigating how much resemblance snapshots of the host graph based on previous time intervals bear to the current one. Given a weighted network representation of the host graph, where an edge $w_{ij}(t)$ stands for the number of clicks from host i to host j in time interval t, we can quantify this resemblance by defining *generalized temporal precision* and *generalized temporal recall* based on true positive click predictions.

We first define generalized temporal precision P as

$$P(\delta) = \left\langle \frac{\sum_{ij} \min[w_{ij}(t), w_{ij}(t-\delta)]}{\sum_{ij} w_{ij}(t-\delta)} \right\rangle_{t \in [\delta, T]}$$

where the averages $\langle \cdot \rangle$ run over all eligible pairs of hourly snapshots and T is the total number of snapshots available in the data set. This measure reflects how many of the clicks present in the older snapshot are also present in the newer one. If the newer snapshot is a superset of the older one, then $P = 1$.

Similarly, we define generalized temporal recall R as

$$R(\delta) = \left\langle \frac{\sum_{ij} \min[w_{ij}(t), w_{ij}(t-\delta)]}{\sum_{ij} w_{ij}(t)} \right\rangle_{t \in [\delta, T]}$$

. This measure reflects how many of the clicks present in the newer snapshot were also present in the older one. If the newer snapshot is a subset of the older one, then $R = 1$.

I calculated these measures over $\delta_M T$ comparisons of hourly snapshots of the weighted host graph, where $\delta_M = 168$ hours (one full week) is the maximum delay considered and $T = 4,996$ is the total number of hourly snapshots available. The baseline for these measures is stationary traffic: if the weighted host graph does not change at all, we have perfect predictability, and $P = R = 1$ for any value of δ.

The results of this analysis are shown in Figure 4.2, which plots generalized temporal precision and recall versus delay for clicks in the HUMAN data set. As one would expect, predictability decays fairly rapidly; however, both precision and recall are quite high (above 50%) for $\delta \leq 3$ hours. We can also observe very strong daily and weekly cycles. After more than four hours have passed, the requests from the prior day at the same time yield higher precision and recall. Even after going back two or more days, it is possible to predict more than 40% of the clicks, which yields better performance than using data from only ten to twelve hours earlier.

Startlingly, we observe a large volume of stationary traffic, as suggested by the fact that P and R never fall below 32%. In other words, almost a third of all requests are from the same sites, to the same sites, every hour of every day. The fact that precision and recall track each other so closely is further evidence of this stationary traffic. For example, 47% of clicks at any given time are predicted by the clicks from the previous day at the same time, and the same percentage are repeated the next day at the same time. This also highlights the difficulty of finding obvious trends in the data, since

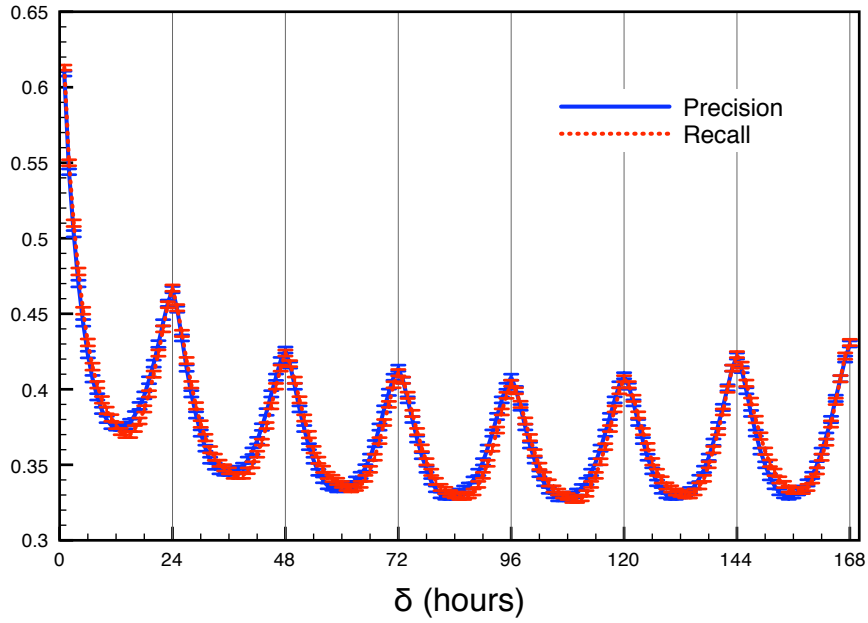


Figure 4.11: Average temporal precision and recall as a function of delay δ for the HUMAN data set. The error bars correspond to ± 1 standard error.

most of the hourly data sets are predicted by a previous set about as accurately as they predict a future set.

The results for the FULL data set, which are not shown here, exhibit almost identical features, other than a uniform increase of about 10% in both precision and recall. This is likely to be an artifact of commonly embedded files such as style sheets and images, which will generally be represented as a click from a host to itself. Because the FULL data set includes automated crawling traffic, it is also possible that such traffic is more predictable.

The baseline prediction that traffic will remain static is obviously the most naive possibility. Intuitively, there ought to be features of the topology of the host graph and observable trends in the weights that allow for significant improvements on the steady-state prediction. If such an improvement is possible, it is more likely to involve topological analysis than simple forecasting based on

the weights. I have experimented with a wide variety of methods for weight forecasting, ranging from cubic spline interpolation to weighted averages, and in no case have I been able to achieve performance statistically significantly better than the baseline. A combination of topological analysis and trending of the weights might be a promising avenue for future research.

Rethinking the random surfer model

The availability of a traffic-weighted version of the Web host graph makes it possible to investigate whether the random surfer model that underlies the PageRank algorithm actually describes patterns of Web traffic. Simply stated, how good is PageRank as a model of Web navigation? This question can be made concrete by considering the primary role of PageRank as a means of ordering search results: how well does the ranking of Web sites produced by PageRank approximate that obtained from the traffic of actual Web users?

The answer to this question is a matter of considerable relevance. Content-independent ranking is a critical necessity for search engines, so that the most important pages that match a query can be brought to the user's attention. PageRank's importance also goes beyond its application to Web search; this topological network measure remains a key tool for researchers studying the structure of large information networks, the Web being the foremost example among many. PageRank serves as the reference model for the dynamic behavior of the multitudes of people who forage for information in these complex networks.

The PageRank metric has several interpretations, ranging from linear algebra to spectral theory, and many implementation issues. The focus here is solely on the intuitive interpretation of PageRank as the stationary distribution of visit frequency by a modified random walk on the Web link graph—in other words, as a simple model of the traffic flow produced by Web navigation.

Formally speaking, PageRank is the solution of a massive system of linear equations:

$$PR(j) = \frac{\alpha}{N} + (1-\alpha) \sum_{i:w_{ij}\neq 0} \frac{PR(i)}{k_{out}(i)}$$

where N is the number of nodes (such as Web pages, or in the present discussion, Web sites); and the parameter α is a teleportation factor, often referred to as a "damping" or "jumping" factor. The first term describes the process by which a user stops browsing at random nodes and teleports (jumps) directly to some other random node without following an edge. The second term describes a uniform random walk (surfing) across links, with the sum running over the incoming links of node j.

The parameter α models the relative probability of surfing versus teleporting and is necessary to avoid becoming stuck in pages with no links ($k_{out} = 0$). We have already seen that the empirical data support a much higher teleportation probability than the customary value $\alpha = 0.15$ [29]: $\alpha \approx 0.54$ for human browsers, or $\alpha \approx 0.2$ even when crawlers are included. A number of studies have explored the role of α in influencing the PageRank algorithm [25, 66]; for the present PageRank calculations, it suffices to simply use the customary value $\alpha = 0.15$.

Aside from the teleportation factor, the interpretation of PageRank as a model of graph navigation is based on three fundamental assumptions implicit in the definition above:

1. An equal probability of following each link from any given node: $\forall i, j : \Pr(i \to j|\text{click}) = \begin{cases} 1/k_{out}(i) & w_{ij} > 0 \\ 0 & \text{otherwise;} \end{cases}$;
2. An equal probability of teleporting *to* each of the nodes in the graph: $\sum_i \Pr(i \curvearrowright j|\text{jump}) = 1/N$; and
3. An equal probability of teleporting *from* each of the nodes: $\sum_j \Pr(i \curvearrowright j|\text{jump}) = 1/N$.

The weighted host graph makes it possible to perform an indirect validation of these assumptions: we can simply compare the traffic predicted by running the PageRank model against the host graph, with the actual traffic flows generated by users at Indiana University, as captured by the in-strength $s_{in}(j)$ for each host j.

Since the primary application of PageRank to rank sites, it is appropriate to focus on the ranking produced by PageRank rather than the actual values of PageRank vector. To compare the rankings of Web sites according to two different criteria (e.g., PageRank and actual traffic), we can use the established Kendall's τ rank correlation coefficient [89]. This measure is intuitively defined from the fraction of pairs whose relative positions are concordant in the two rankings:

$$\tau_b = \frac{4C}{N(N-1)} - 1$$

where C is the number of concordant pairs and the subscript b (dropped henceforth) refers to the method of handling ties. The values of Kendall's τ range from 1 in the case of perfect agreement, down to -1 is the case of perfect inversion, with $\tau = 0$ indicating the absence of correlation. At first glance, Kendall's τ would thus seem to require $O(N^2)$ comparisons, but it can be computed efficiently using Knight's $O(N \log N)$ algorithm in the manner implemented by Boldi *et al.* [24].

Figure 4.2 plots the rank correlation for subsets of θ top-traffic hosts. In other words, when $\theta = 1,000$, we are comparing a list of the 1,000 hosts with the most traffic to PageRank's ordering of the same hosts. Consider the plot for the HUMAN data set, which shows the correlations from the small set of most popular sites all the way to the entire network of four million hosts. In particular, consider the correlation between the ranking estimate by PageRank (PR) and that obtained from empirical traffic data (s_{in}). Toward the end of the trace, we can see that as more low-traffic hosts are added, the correlation increases up to almost 0.7. This is not terribly meaningful, however, since the

traffic data are quantized and low-traffic sites dominate the bottom of the distribution, producing a large number of ties that drive up the correlation measure.

For the top sites, however, the correlation is quite weak ($\tau < 0.2$ up to a million hosts or so). This is consistent with findings based on earlier traffic data from the Polish Web [139]. The conclusion is surprising: PageRank is actually a poor predictor of traffic ranks for the most popular portion of the Web, and the least popular portion is so infrequently visited that we cannot properly evaluate it.

A reasonable hypothesis is that that poor performance on the part of PageRank results from the assumptions outlined above. Of those three assumptions, the first one (that there is an equal chance of following any link from a page) can be easily removed by ranking the hosts according to a third, intermediate measure between empirical traffic and PageRank. Let us define *weighted PageRank* PRW by incorporating the empirical link weights into the PageRank expression

$$PRW(j) = \frac{\alpha}{N} + (1-\alpha) \sum_{i:w_{ij}\neq 0} \frac{w_{ij}}{s_{out}(i)} PRW(i).$$

The performance of this weighted PageRank measure is also shown in Figure 4.2. It performs only slightly better as a predictor of traffic, and is much more correlated with the original PageRank than either are correlated to actual traffic. The results of this experiment suggest that if PageRank's problems have to do with its assumptions of uniform distributions, those errors are dominated by violations of the latter two assumptions, which have to do with teleportation.

To better understand how all of these assumptions affect the ranking of Web sites, we can consider each hypothesis in turn and attempt to quantify the degree to which it is supported by empirical data.

Assumption 1 deals with the local homogeneity of link weights. We have already seen in the

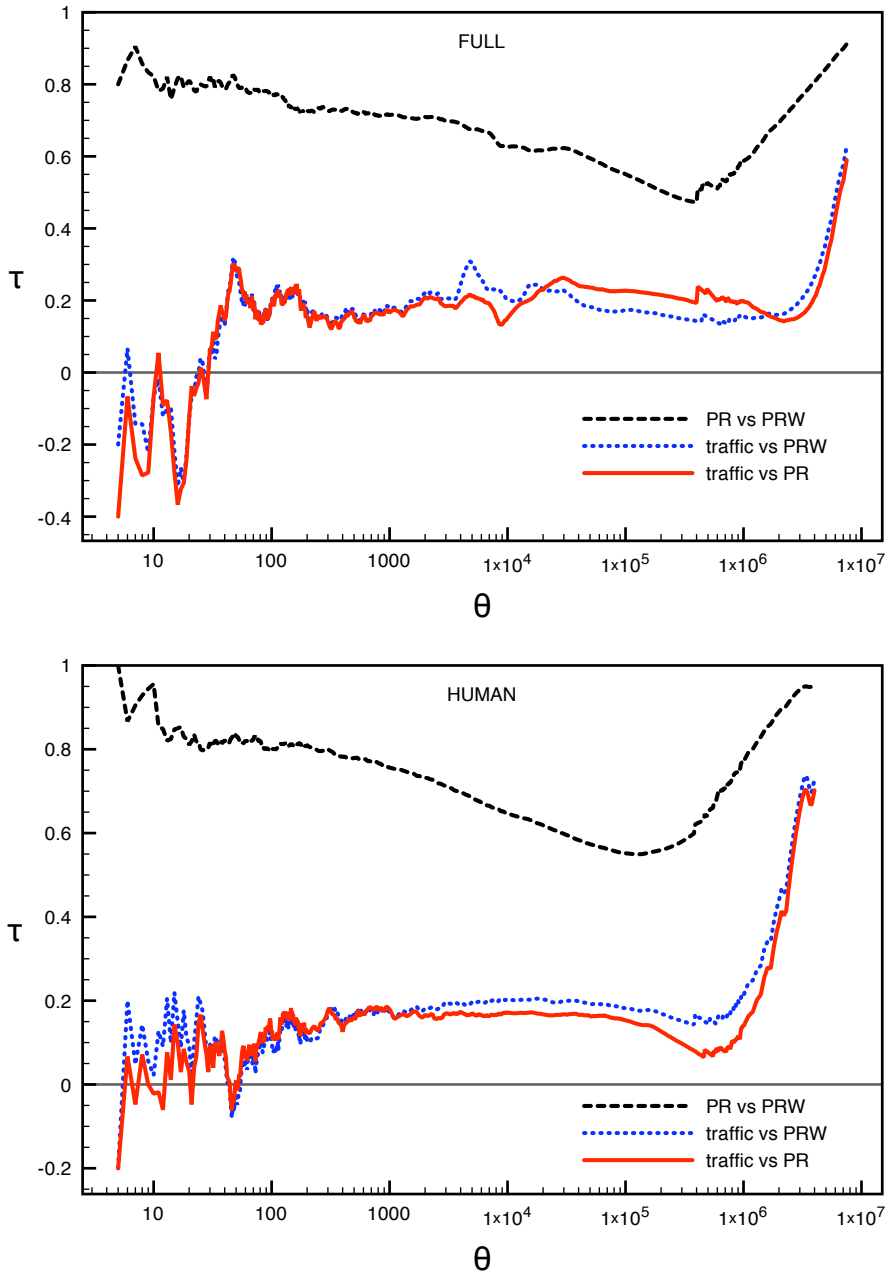


Figure 4.12: Kendall's τ correlations between different rankings of the sites in the FULL (top) and HUMAN (bottom) versions of the host graph, plotted against the traffic rank threshold θ.

previous discussion that link weights are extremely heterogeneous on a global level; this time, the focus belongs on the links from each individual node, i.e., whether surfers are truly equally likely to click on any of the links found on a site. Strong local heterogeneity implies that only a few links carry away the biggest proportion of a site's traffic. Such a heterogeneity would defined specific pathways within the host graph that account for most of the total traffic.

In order to assess the amount of homogeneity at the local level, for each host i, we can calculate the quantity

$$Y_i = \sum_j \left(\frac{w_{ij}}{s_{out}(i)} \right)^2$$

. This function Y_i is the Herfindahl-Hirschman index first mentioned in Chapter 2. Recall that Y_i as a function of out-degree $k_{out}(i)$ characterizes the level of local heterogeneity among the links from i. If all of the weights on the links emanating from a node are equal, the quantity $k_{out}Y(k_{out})$ will be constant with respect to k_{out}, whereas it will grow with k_{out} if the local traffic is extremely heterogeneous and a few links dominate the rest. Increasing deviations from constant behavior therefore indicate local heterogeneity, meaning that traffic from a site is focused on a small number of links, with the remaining edges carrying just a small fraction of the traffic.

The distribution of k_{out} versus $k_{out}Y(k_{out})$ is shown in Figure 4.2. The fit shows that actual traffic follows a scaling law $k_{out}Y(k_{out}) \sim k_{out}^{\lambda}$ with $\lambda \approx 0.8$. This represents an intermediate behavior between the two extremes of perfect homogeneity ($\lambda = 0$ if all links are equal) and perfect heterogeneity ($\lambda = 1$ if all traffic is concentrated on a single link). The results are thus consistent with the existence of dominant pathways in the host graph: most traffic entered a site from its major incoming links and leaves it through its major outgoing links, as already suggested by Figure 4.2.

However, some degree of local heterogeneity can be expected, given the broad distribution of

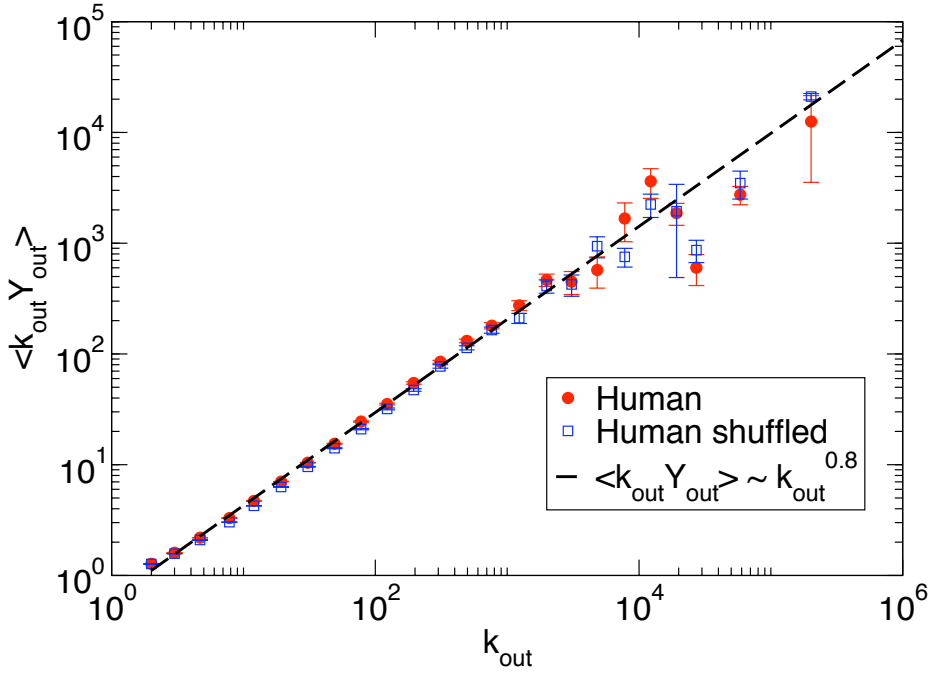


Figure 4.13: Mean behavior of $k_{out}Y(k_{out})$ versus k_{out} for the HUMAN host graph. $\langle Y(k_{out})\rangle$ values are obtained averaging Y_i across nodes i grouped into logarithmic bins by out-degree. The error bars correspond to ± 1 standard error on the bin averages. Also shown is the same measure for a shuffled version of the HUMAN host graph in which the link weights have been randomly reassigned across all edges.

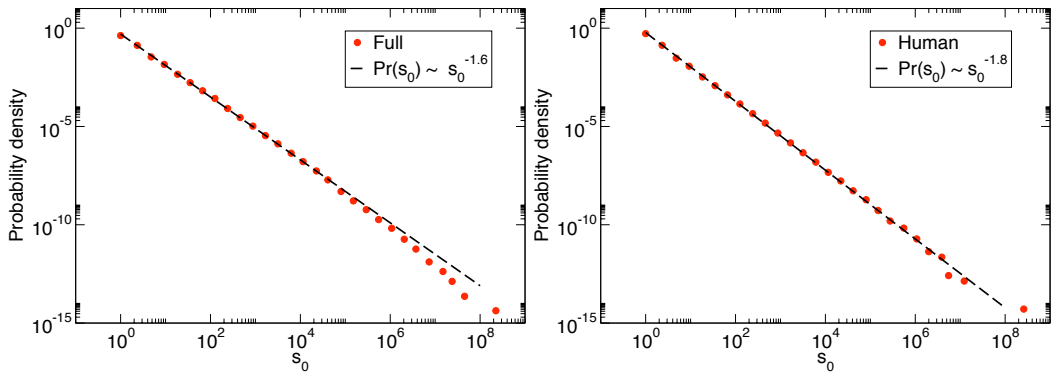


Figure 4.14: Distribution of requests with empty referrer for the FULL (left) and HUMAN (right) host graphs.

weights seen in Figure 4.2. We can test whether the weight distribution is responsible for the apparent local heterogeneity of weights by shuffling the weights randomly and repeating the analysis. Figure 4.2 shows that the same scaling behavior for $k_{out}Y(k_{out})$ is observed even when the weights are shuffled, suggesting that the local heterogeneity in link weights is indistinguishable from accidental local correlations among highly diverse weights. This suggests a basic interpretation of the observed correlation between traditional and weighted PageRank in Figure 4.2: local weight diversity simply does not explain much of the difference between PageRank and actual traffic.

Assumption 2 concerns homogeneity of teleportation destinations: PageRank assumes that all sites are equally likely to be the starting points for a sequence of page visits. This assumption can be tested using the empty referrer node in the traffic-weighted host graph. For each host i, let $s_0(i)$ be the number of jumps to i, i.e., the number of requests that have an empty referrer and i as the target. Once normalized, this is a direct measure for the probability that a site is a starting point for Web surfers. Figure 4.2 plots the distributions of s_0 for the FULL and HUMAN data sets, revealing a very broad power-law distribution with an exponent between 1.6 (for the FULL data set) and 1.8 (for the HUMAN data set).

Once again, we have an exponent below two, implying that both the variance and the mean

of the distribution diverge in the limit of large graphs and are bounded only by the finite size of the data actually gathered. These results strongly violate PageRank's assumption of homogeneous teleportation, which would manifest itself in a narrow distribution and helps to explain the low correlation between PageRank and actual traffic. These results also make intuitive sense: people are much more likely to begin surfing at a few very popular sites (search engines, news sites, etc.) than to the great majority of other sites.

Assumption 3 is about the homogeneity of teleportation sources. In the PageRank model, all sites are equally likely to be the end-points of sequences of surfing clicks; the random surfer is equally likely to start somewhere else regardless of their current position. Once again, this assumption is testable by the data, though we must be careful of our methodology.

For each host i, $s_{in}(i)$ is the number of arrivals of a surfer at i (requests having i as the target) and $s_{out}(i)$ is the number of departures of a surfer from i (requests having i as the referrer). However, the strength differential $s_{in}(i) - s_{out}(i)$ is not the same as the number of paths that have terminated at i, because multiple paths can start from i. For example, this happens when users hit the back button or follow multiple links in different browser tabs; cached pages do not generate new requests. This means that the very nature of traffic data does not allow us to validate this assumption directly. However, we can use the ratio $s_{out}(i)/s_{in}(i)$ to measure the likelihood that traffic into i leaves i by clicking on links from i. This "hubness" measure is not a probability; in fact, we can have $s_{out}(i)/s_{in}(i) \gg 1$ because of multiple traffic paths from i. Even so, the larger the ratio $s_{out}(i)/s_{in}(i)$, the more likely i is to be a starting hub, and the less likely it is be a starting sink (teleportation source).

The distributions of s_{out}/s_{in} for the FULL and HUMAN data sets can be seen in Figure 4.2. We can observe a very broad distribution, with an initial plateau in the regime where $s_{out} < s_{in}$, followed by a power law decay for the regime where $s_{out} > s_{in}$. The central peak corresponds to sites

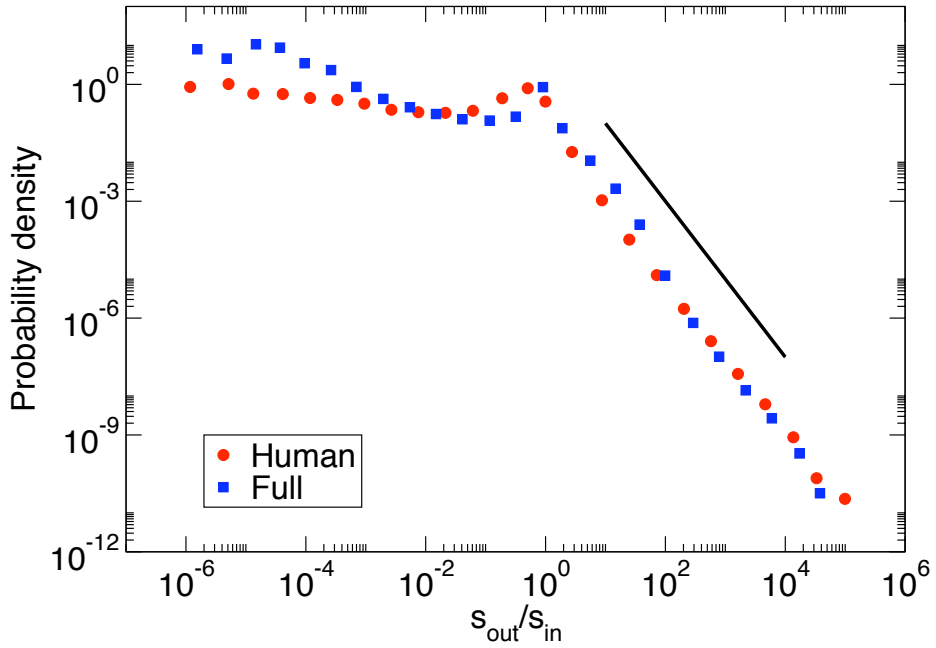


Figure 4.15: Distribution of the ratio of outgoing strength s_{out} to incoming strength s_{in} for the FULL and HUMAN data sets. This traffic ratio is very large for popular hubs from which users follow many links. A power law model with exponent 2 is included as a visual reference.

where traffic is conserved; that is, $s_{out} = s_{in}$. While this result is not a direct check of the validity of the third assumption, it does show that people follow many more links from a relatively small number of popular hubs that from the great majority of less popular sites, helping to further explain the low correlation between PageRank and actual traffic rankings. The two clearly demarcated regimes suggest the possibility of using the s_{out}/s_{in} ratio as a topology-independent criterion for identifying hubs. A promising avenue of future research would be to investigate whether Kleinberg's HITS algorithm [90] is able to predict these strong hubs.

The results in this section show that a large fraction of traffic is driven by teleportation rather than hyperlinks. Real users often begin browsing from bookmarks, default home pages, or external applications, and this process is not captured well by uniform random jumps. This raises the important question of how to better model teleportation: what *are* the preferred starting points of our

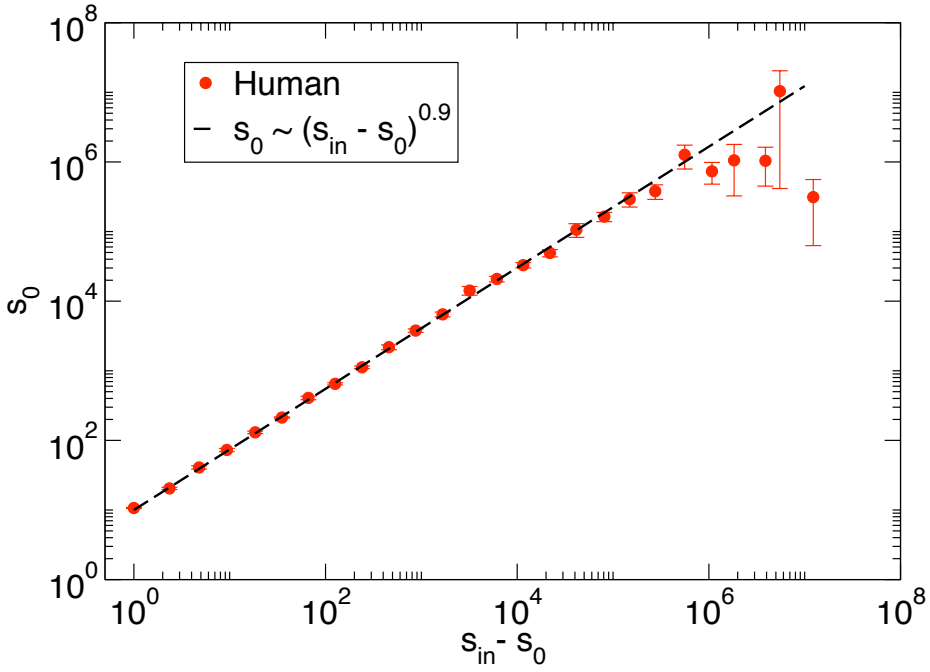


Figure 4.16: Correlation between traffic from the empty referrer and traffic from navigation in the HUMAN host graph. For better visualization, values of s_0 are averaged within logarithmic bins in the $s_{in} - s_0$ axis. The error bars correspond to ± 1 standard error on the bin averages.

navigation on the Web? The analyses above tell us that preferences do exist and are so strong as to lead to scale-free distributions, but they say nothing about what those preferences actually are.

We can gain a feel for the answer to this question by seeing whether the probability of starting from a site is correlated with the probability of arriving at the same site through navigation. Figure 4.2 shows that there is indeed an extremely strong correlation between traffic to a site through navigation ($s_{in} - s_0$) and traffic to the same site from the empty referrer (s_0). In fact, the two are almost linearly related. The pages from which people start their browsing tend to be the same as those where they are likely to end up: there is a single notion of popularity.

4.3 Web sessions

The findings described thus far on the structure and dynamics of the host graph were both intriguing and inspiring to me, especially the quantification of the mistaken assumptions of the random surfer model. Clearly, the standard model fails to reflect the surfing process actually followed by real users. However, the combination of the inability to capture all requests and the disposal of all distinguishing data about client made it difficult to proceed with further analysis of these ideas. In order to fix the random surfer model, we need a better idea of what *individual* surfers do and how their activity combines to produce the phenomena discussed in the previous section. The undifferentiated click data does not make it possible even to typify the mean user, since we cannot know either how many users contribute to the data set or how many requests were missed.

Fortunately, an opportunity arose to gather an additional data set in which client identities were preserved, allowing me to extend this research into investigating the individual behavior of users, their browsing sessions, and ultimately, as we shall see in Section 4.4, modeling that behavior in a plausible way that compares favorably to real traffic data. In this section, I first describe this user-based data set: the means of collection, its basic properties, and its dimensions. I then describe the validation of the data through repetition of some of the host-based analysis before characterizing the properties of individual users. Finally, I describe the results of segmenting the click streams of individual users into sessions based on several criteria, leading to a new definition of Web session that captures user interaction in a more robust way than simple activity timeouts.

Dormitory click data

The user-based data set was gathered in collaboration with Dr. Jean Camp of the School of Informatics and her student J. Duncan in connection with their study of user homophily in Web

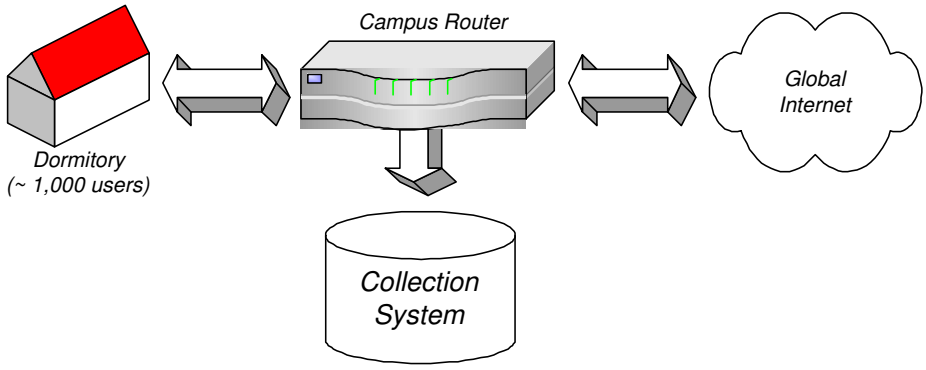


Figure 4.17: System architecture for the collection of user-based click data.

data. The users concerned were the residents of Read Hall, one of the undergraduate dormitories on the Bloomington campus. This residence hall consists of four wings of five floors each and is home to just over a thousand undergraduates, split roughly evenly between men and women. Because of its location, this building has a somewhat greater proportion of music and education students than other campus housing, but it cannot be considered atypical of residential housing on an American college campus.

The architecture for collecting this data, shown in Figure 4.3, is much the same as for the university-wide data. The collection system is once again a dedicated FreeBSD server, and the collection software is nearly identical but for its ability to retain the MAC address of each client system. The MAC address is used only in order to distinguish the traffic of individual users. This involves an assumption that most computers in the building have a single primary user, which is reasonable in the light of the connectedness of the student population and the small number of public workstations available in the building. Furthermore, as long as users do not replace the network interface in their computers, this information remains constant—far more so than would be the case for IP addresses.

Because this data set involves the retention of distinct client identities, it was subject to an agreement with the Human Subjects Committee. The students were all informed of the study and the conditions under which the data would be gathered, and we established tight restrictions on access to the collection system, how the raw data would be stored and handled, and the conditions under which information derived from the data could be shared with other researchers. No individual other than myself has ever had access to the actual MAC addresses used by the users' machines; anonymization of this information is the very first step of the analytical process.

The aggregate traffic of the dormitory was sufficiently low so that our tuned sniffing system could maintain a full rate of collection without dropping packets. This data set thus offers the twin advantages over the university-wide data set that it not only divides clicks among actual users, but divides *all* of their clicks. All of the other caveats associated with the university-wide data do still apply, and it must be kept in mind that this time we have two orders of magnitude fewer users, and less diversity among those users. Nevertheless, such a data set represents a rare opportunity to perform the sort of analysis described here, for which I am grateful.

The user-based click data was collected over a period of about two months, from March 5, 2008, through May 3, 2008. This period included a week-long vacation during which no students were present in the building. During the full period of data collection, the server logged nearly 408 million HTTP requests from a total of 1,083 unique MAC addresses.

Recall from the previous section that only a minority of HTTP requests actually indicate an actual human being trying to fetch a resource corresponding to a Web page, leading to the creation of the FULL and HUMAN data sets. In this analysis, I retain only the requests corresponding to the HUMAN data sets; that is, I save only those URLs that are likely to be requests for actual Web pages, as opposed to media files, style sheets, Javascript code, images, and so forth. The motivation behind this filtering is that most of these other resources are included by actual Web pages; in general,

browsers will fetch these resources automatically without any user direction or intervention. They can therefore contribute little to our understanding of the behavior and motivation of users.

I also filtered out a small subset of users with negligible activity; their traffic consisted almost entirely of automated Windows Update requests, which similarly provide no meaningful data about user-directed activity. They simply owned a computer that happened to be plugged in. Finally, I also discovered the presence of a poorly-written anonymization service that was attempting to obscure traffic to a particular adult chat site by spoofing requests from hundreds of uninvolved clients. Because these requests were not genuinely generated by the MAC addresses indicated by the captured packets, they were also removed from the data set.

Initial analysis revealed that some Web clients issue duplicate HTTP requests (i.e., with the same referring and target URLs) in nearly simultaneous bursts. These bursts appear to occur independently of the type and domain name of the URL being requested, and are less than a single second in duration. They may simply involve checking for updated content, but it is impossible to confirm this without access to the original HTTP headers (and possibly the server responses). Because this behavior occurs so rapidly that it cannot reflect the deliberate activity of individual users, I also filtered these duplicate requests from the data.

Finally, the agreement with the Human Subjects Committee obliged us to try to remove all individually identifying information from the referring and target URLs. Because usernames (and in too many cases, passwords as well) are often stored in CGI variables embedded directly in URLs, the most effective means of doing so was to strip off all identifiable query parameters from the URLs. Applying this anonymization procedure affects roughly one-third of the requests remaining after the filtration process.

The resulting data set, summarized in Table 4.3, is the basis for all of the description and analysis that follows.

Table 4.3: Approximate dimensions of the filtered and anonymized user-based data set.

Page requests	29.8 million
Unique users	967
Web servers	630,000
Referring hosts	110,000

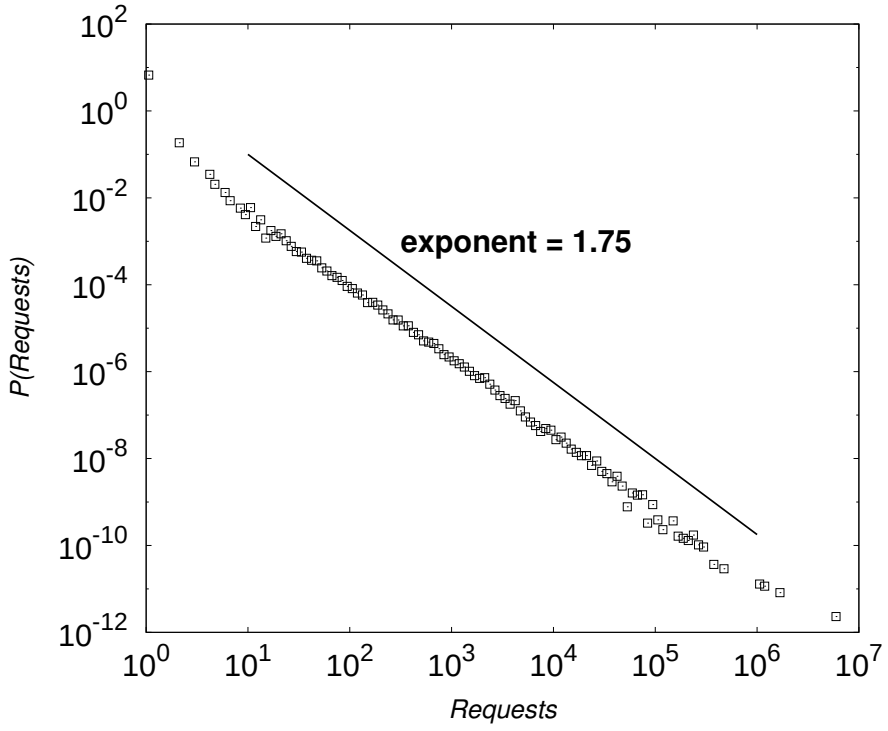
Host-based properties

The first priority in analyzing this data set was to verify that its statistics were consistent with those of previous studies. In the study just described, we saw that the distribution of the number of requests directed to each Web server (s_{in}) could be well-fitted by a power law $\Pr(s_{in}) \sim s_{in}^{-\gamma}$ with exponent $\gamma \approx 1.8$. Figure 4.3A shows that the distribution of s_{in} in the present data set is consistent with this: the distribution is linear on a log-log scale for nearly six orders of magnitude, with a slope of roughly 1.75. Similarly, for the university-wide study, we saw that the number of requests citing each Web server as a referrer (s_{out}) could be approximated by a power law with $\gamma \approx 1.7$. In the present study, we find that $\gamma \approx 1.75$ for the distribution of s_{out}, as shown in Figure 4.3B. The overall distribution is thus found to be in concordance with the prior results.

An examination of assortativity in the case of this data set, shown in Figure 4.3, is also consistent with the previous study. We again find that the degree patterns in the graph are disassortative, consistent with a technological network, but the superimposition of user behavior moderates this effect.

Whereas the previous study was conducted under conditions of complete anonymity for users, this one does attribute each request to a particular user. This makes it possible to extend the investigation of host-based properties by examining the relative popularity of Web servers as measured by the number of distinct users contributing to their traffic. The results of this analysis are shown in Figures 4.3A and 4.3B. We find that the distribution of the number of users u contributing to the

A



B

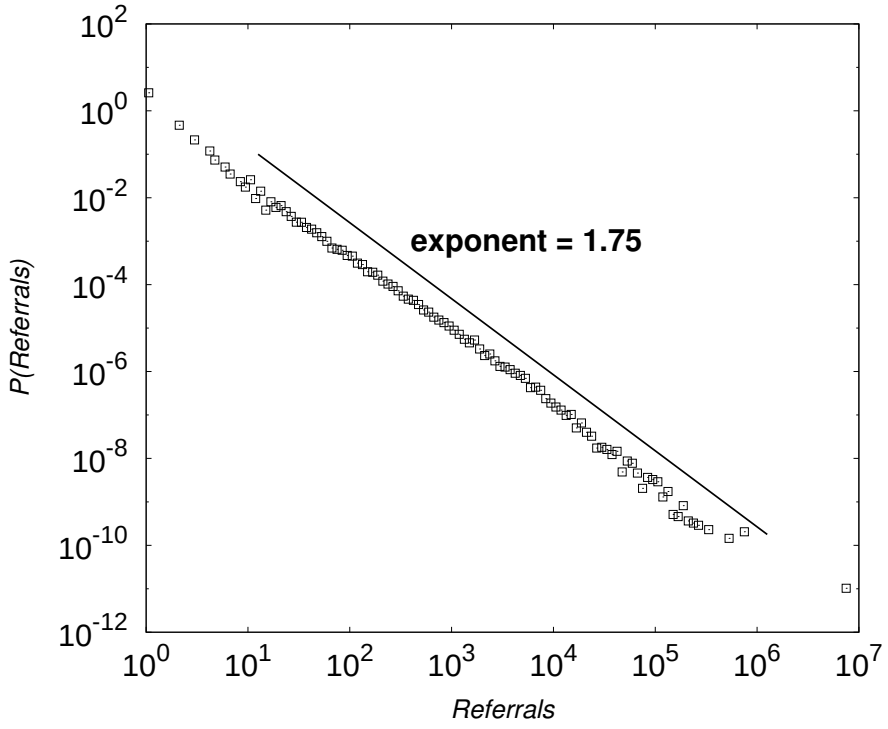


Figure 4.18: Distributions of in-strength (A) and out-strength (B) for each Web server in the user-based data set.

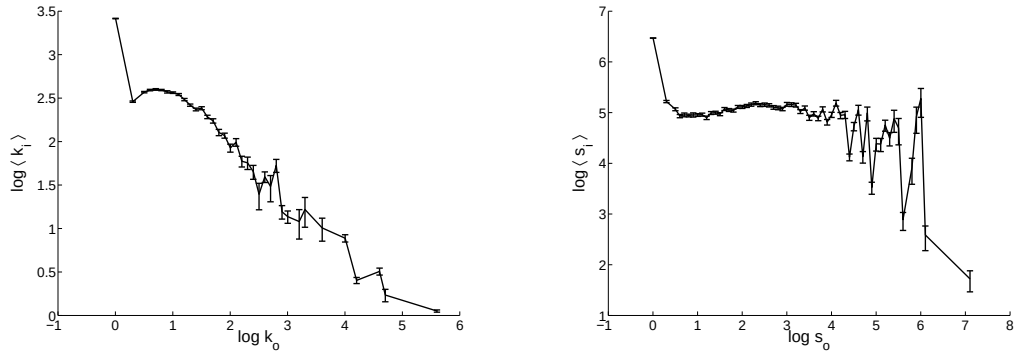


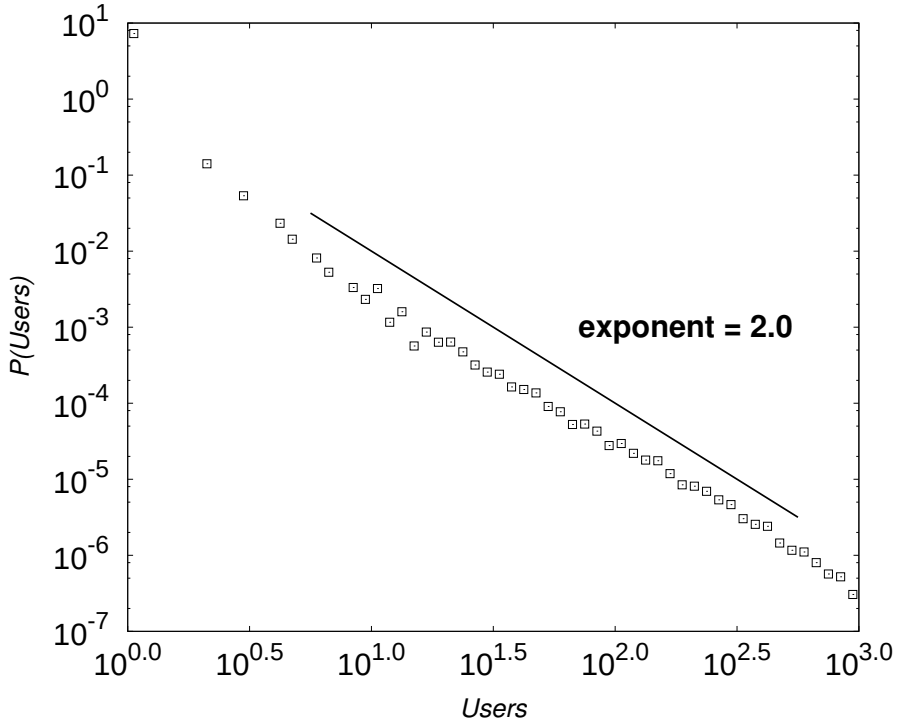
Figure 4.19: Assortativity in the host graph for the user-based data set, examined in terms of degree (left) and strength (right). Once again, each plot shows the in-degree [in-strength] of nodes as a function of the out-degree [out-strength] of the nodes linking to them.

inbound traffic of a Web server is well-approximated by a power law $\Pr(u) \sim u^{\beta}$ with $\beta \approx 2.0$ and the outbound by a power law with $\beta \approx 1.9$.

As in previous cases, these exponents require some further comment. In what is becoming a familiar situation, we see it seems that $\beta \leq 2$ in the case of both incoming and outgoing popularity of Web servers, implying that these distributions lack any intrinsic mean. This suggests a lack of any inherent ceiling on the popularity of a Web site, regardless of the size of the user population. Indeed, further analysis of the data shows that the social networking site Facebook is a popular destination for almost 100% of the students in the study, handily eclipsing any major search engine or news site.

The limited size of this data set offered me the first opportunity to investigate the distribution of clustering coefficients of the nodes. The results, however, turned out not to be compelling: while the mean clustering coefficient is extremely high relative to that of a random network (around 0.32 when the network is considered at the level of individual URLs), the distribution offers few compelling insights. Further inspection reveals significant peaks for every possible ratio of small integers, highlighting a problem with most definitions of clustering: the degree of the node is not

A



B

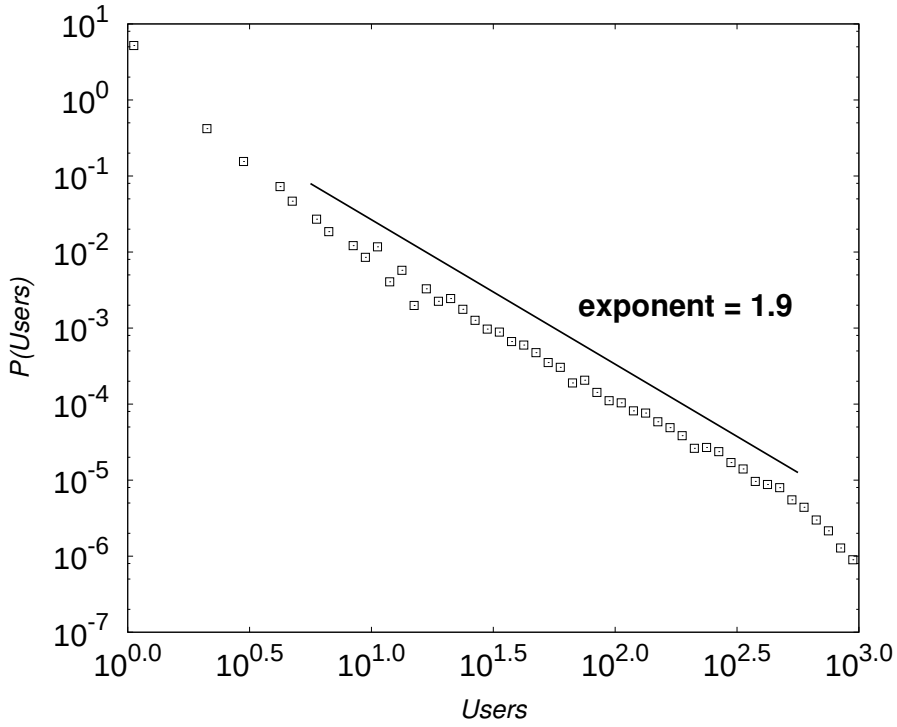


Figure 4.20: Distributions of the number of unique users incoming to (A) and outgoing from (B) each Web server in the data set. These distributions serve as a rough measure of the popularity of a Web site and imply that the potential audience of the most popular sites is essentially unbounded.

taken into account, even though full connectivity among the neighbors of a node with $k = 20$ is far more interesting and unlikely than full connectivity among the neighbors of a node with $k = 2$.

User-based properties

Having established the consistency of the data set with prior results and the heavy-tailed distribution of popularity among Web sites, we can finally shift our focus to the individual user. The behavior of individual Web surfers is of critical interest for not only models of traffic, but also a variety of applications such as network anomaly detection and the design of anonymization tools. Because nearly all of the per-server distributions in Web traffic exhibit scale-free properties and have extremely long and heavy tails, one might anticipate that the same would be true of Web users. The implications of this are serious: if the statistics that describe user behavior lack well-defined central tendencies, then very little individual behavior can be described as anomalous. However, simple biology dictates that any given user has only a limited amount of time and energy to devote to Web surfing, so we know that user-based distribution must be bounded in some way. The key question is whether we can characterize "normal" individual traffic in a meaningful way. If we can establish a clear picture of the typical user, unusual users become easier to identify and have a difficult time maintaining their anonymity.

I began this investigation by considering the distribution of sizes of the users' individual click streams, both in terms of the total number of requests generated by each user and the number of empty-referrer requests generated. The second distribution is interest because it describe the number of times a user has jumped directly to a specific page instead of navigating there from already viewed pages. Recall from the previous section that the random surfer model assumes this to be a uniform distribution.

The actual distributions are shown in Figures 4.3A and 4.3B. Although the smaller size of these

distributions (roughly 1,000 points) makes fitting more difficult than in the case of the enormous per-host distributions, we can observe reasonably strong log-normal fits for these distributions. While the distribution is broad, there is a well-defined central tendency, and we find that the average user generated around 16,000 requests from 2,500 starting pages over the course of two months. In both of these distributions, a small number of users were removed (roughly 50 in each case) because their click streams were very small: under 2,500 requests in all or under 500 start pages. This filtering is appropriate because the vast majority affected were users who did not begin generating any traffic at all until quite late in the study, possibly because of new hardware or the approach of final exams.

Next I examine the distribution of the ratio of the number of empty-referrer requests to the total number of requests for each user. This yields a rough measure of the teleportation parameter α used by PageRank, which assumes a constant value (usually $\alpha = 0.15$) for all users. A strong central tendency would imply that a random surfer has a fairly constant jump probability *overall* even if the chance of jumping varies strongly from page to page, as suggested by the previous study. As shown in Figure 4.3, we do observe a strong fit to a log-normal distribution with a mean of about 15%. Although the distribution is fairly wide, this mean does match remarkably well the jump probability most often used in PageRank calculations.

Besides the number of requests generated by each user, it is instructive to inspect the rate at which those requests are generated. Because not every user was active for the full duration of data collection, this cannot be deduced directly from the distribution of total requests. Figure 4.3 shows the distribution of the number of requests per second for each user generating an average of at least fifty requests per day over the time they were active. We can again observe a reasonable fit to a log-normal distribution with a mean of about 0.0037 requests per second or 320 requests per day.

Finally, the data offer an opportunity to revisit the question of whether browsing activity is

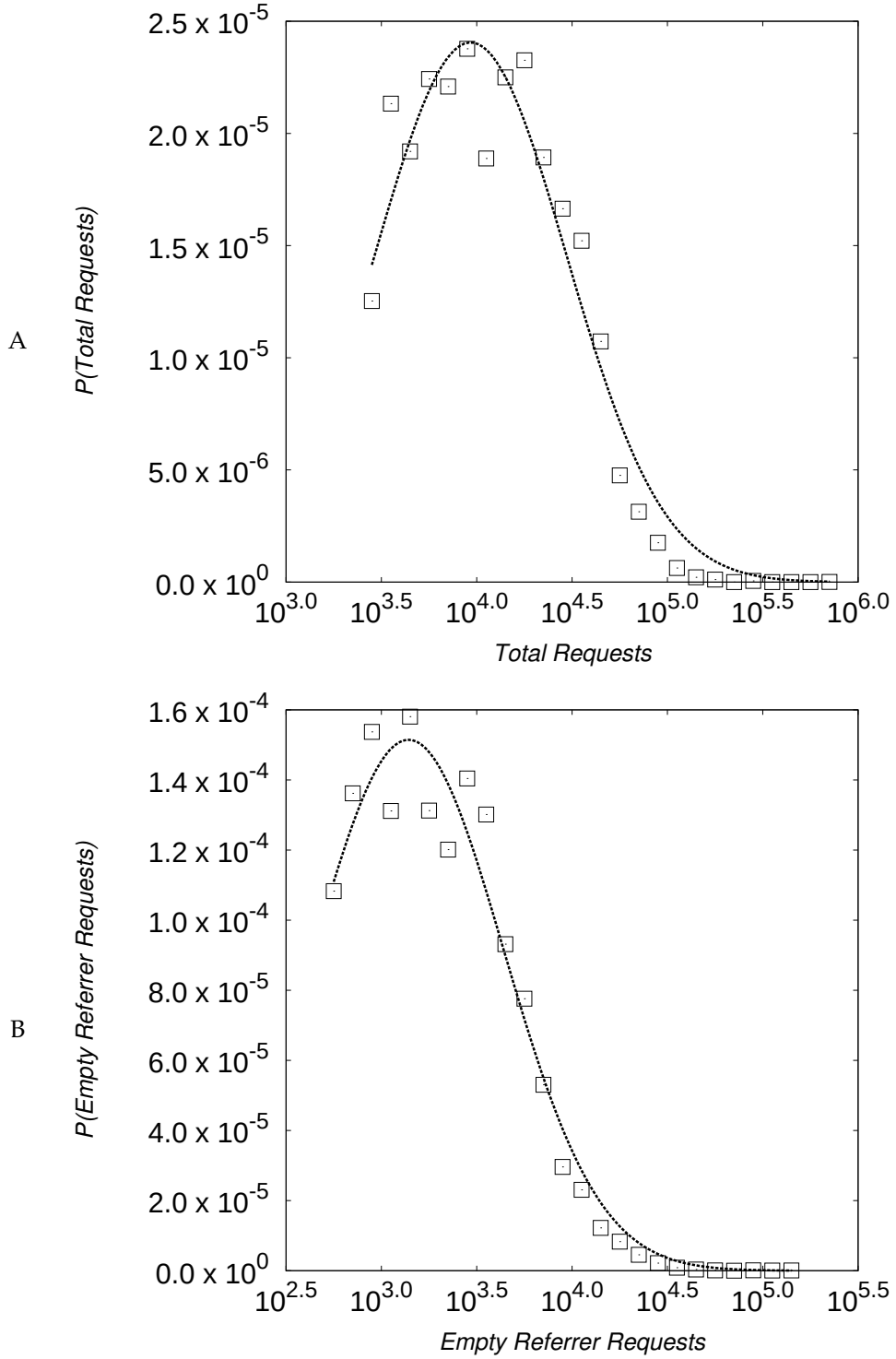


Figure 4.21: Distributions of the number of requests made per user (A) and the number of empty-referrer requests made per user (B). Also shown are reference log-normal fits to each distribution, which omit some low-traffic users, as described in the text.

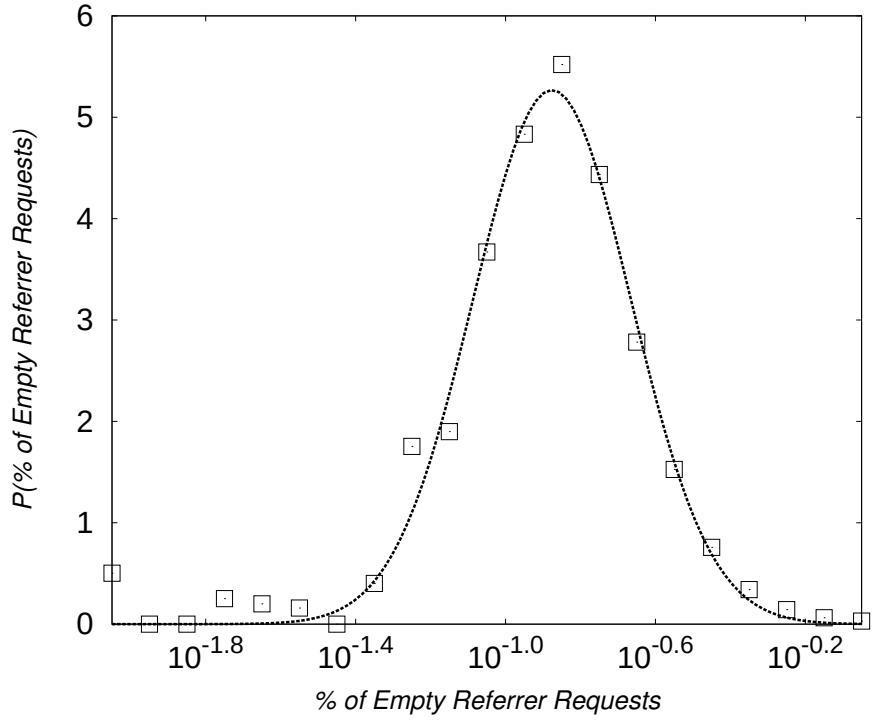


Figure 4.22: Distribution of the proportion of empty-referrer requests made by each user, which roughly corresponds to the chance that a user will jump to a new location instead of continuing to surf. Also shown is a reference log-normal fit to the distribution.

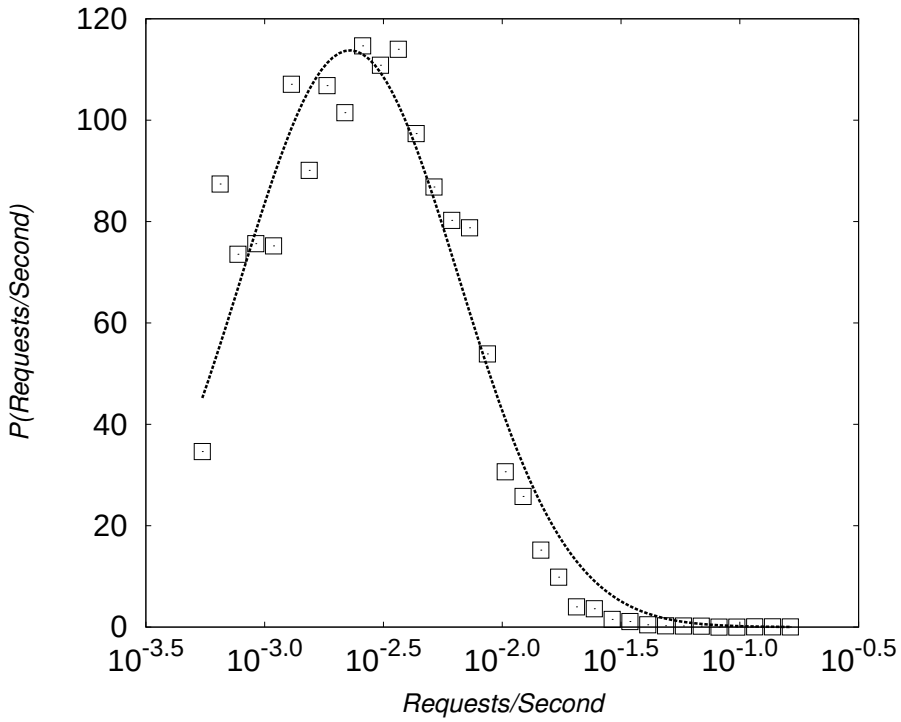


Figure 4.23: Distribution of the number of requests per second made by each user, together with a reference log-normal fit to the distribution.

motivated more by searching or surfing. To do so, let us consider the ratio of the number of unique referring sites to the number of unique target sites for each user. This ratio serves as a rough measure of the extent to which a user's behavior is typified by searching or surfing. If the number of referring hosts is low in comparison to the number of servers contacted, this implies that the user browses the Web through a fairly small number of gateway sites, such as search engines, social networking sites, or a personal bookmark file. Conversely, if the number of referring hosts is high in comparison to the number of servers contacted, this implies that a user is more given to surfing behavior: they tend to arrive at new sites through navigation.

The results are presented in Figure 4.3 and show that the distribution is bimodal and well-fit by the linear combination of two log-normals. This implies the existence of two groups of user: one more oriented toward portals and one more oriented toward browsing. Portal users visit on average almost four sites for each referrer, while surfers visit only about 1.5 sites. In support of this characterization, further analysis reveals that although the overall mean ratio is 0.57, this drops to 0.37 among users who have more than 60% of the traffic connected to Facebook in some way.

Session properties

When we contemplate modeling the behavior of Web users, whether through a simple model like the random surfer or a more complex system, we are naturally drawn to the notion of a Web session. The idea of a session is intrinsic to the design of many Web applications, and the constrained environments in which we are most often able to observe users on the Web make it easy to image that a user sits down at the computer, fires up a Web browser, issues a series of HTTP requests, then closes the browser and moves on to other, unrelated tasks. This is certainly the behavior we observe when users visit a research lab to participate in a study or must dial into a modem pool before beginning to surf the Web. The subjects of the present study did not fit these

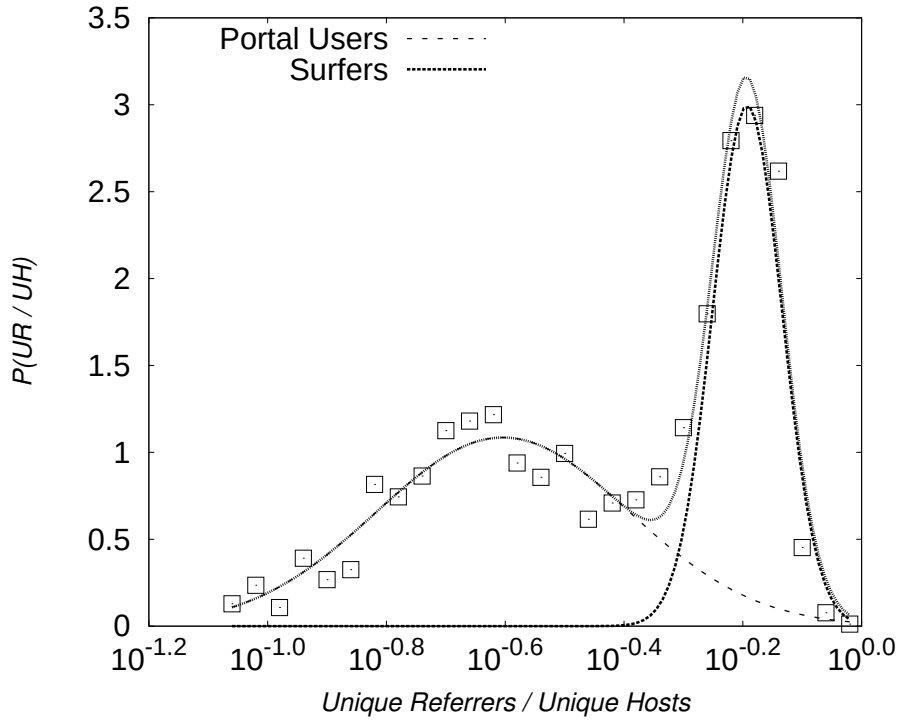


Figure 4.24: Distribution of the ratio of unique referring sites to unique target sites for each user. This bimodal distribution can be modeled by two log-normals with means at 0.28 and 0.65, respectively.

constrained conditions; they have 24-hour access to dedicated high-bandwidth network connections, and the data represent the traffic they generated in an environment that was both their home and their workplace. It seems reasonable that we might face some difficulty in selecting the optimal value for a timeout.

In the first effort of my collaborators and myself to segment individual click streams into sessions, we settled on a five-minute inactivity timeout as a reasonable start, a decision informed by previous research in the field [130]. In the resulting segmentation, each user's click stream split into an average of 520 sessions over the two-month period. A typical session lasted for a bit over ten minutes and included around sixty requests to twelve different Web servers. These values seemed quite plausible for the population: one can easily imagine the typical student participating in ten ten-minute Web sessions every day.

The straightforward approach of identifying sessions using inactivity timeouts thus seemed promising. The next step was to experiment with a variety of different timeouts to find an optimal value. Because of the relatively tractable log-normal distributions of user activity reported above, it did not seem unreasonable to suppose that some of the per-session statistics would remain relatively constant as the timeout was adjusted, and others would show dramatic change in the neighborhood of some critical threshold. Ideally, this critical threshold would prove to be an intrinsic property of human behavior of the Web and vary little from data set to data set.

The results, shown in Figure 4.3, show this to be far from the case. All of the statistics under examination (mean number of sessions per user, mean session duration, mean number of requests, and mean number of hosts contacted) turn out to exhibit strong and regular dependence on the particular timeout used. They show no large discontinuities, areas of particular stability, or even local maxima and minima. This implies there is no reason based on the data itself to select one timeout threshold over any other. The choice is purely arbitrary and it, rather than the notion of a

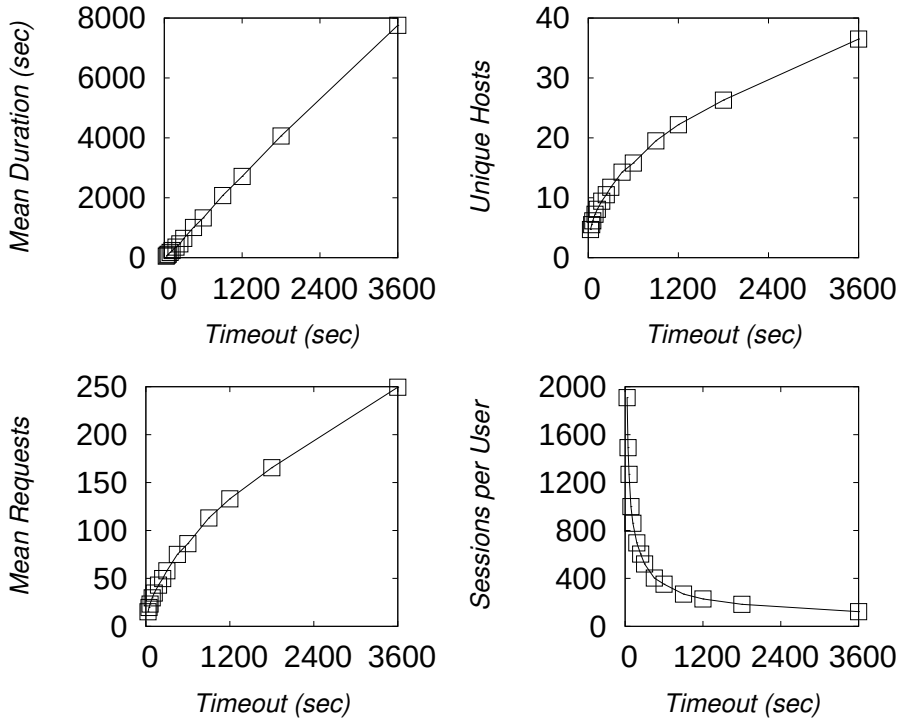


Figure 4.25: Session statistics as a function of timeout in seconds. Top left: Mean duration of sessions in seconds. Top right: Mean number of hosts contacted per session. Bottom left: Mean number of requests per session. Bottom right: Mean number of sessions per user.

session, becomes the prime determiner of all relevant statistics.

While some dependence on the timeout value is only to be expected, this result came as a surprise and led to the conjecture that the observed behavior might be a side-effect of considering every user's sessions as part of the same distribution. In other words, if we were to study the click streams of individual users, we might see more pronounced clustering of HTTP requests in time.

To test this notion, I picked several users at random. For each one, I calculated the distribution of "interclick times," defined as the number of seconds elapsed between each pair of successive page requests. A user with n requests in their click stream would thus yield $n - 1$ interclick times. If a user's activity were typified by tight bursts of requests with long periods of inactivity between

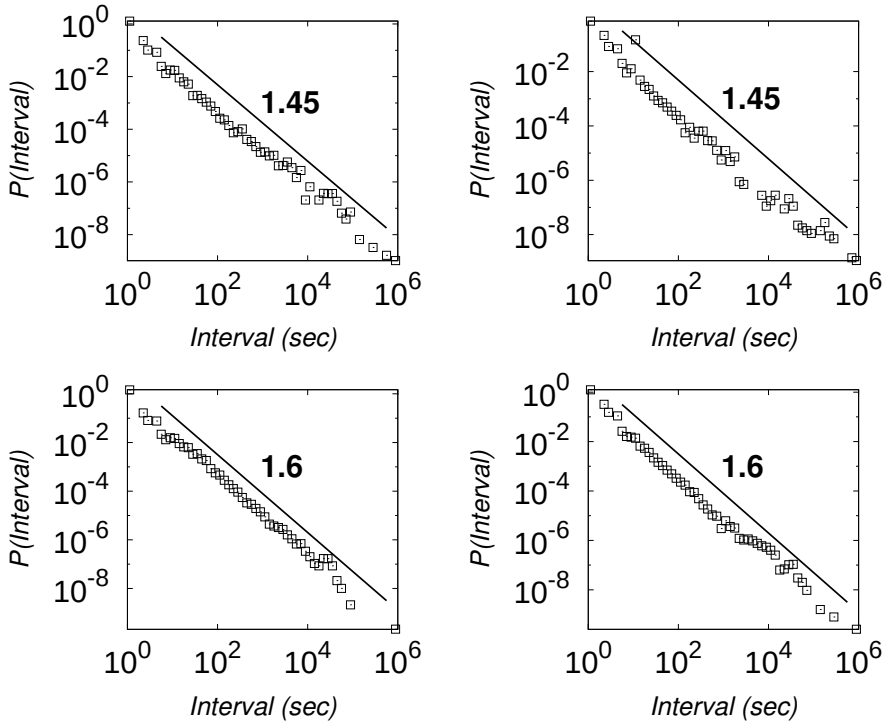


Figure 4.26: The distributions of the time between successive requests for four randomly selected users. Each distribution is well-approximated by power laws with exponents below two, suggesting both unbounded variance and the lack of a well-defined mean.

them, we would expect to find a steep decline in the tail of the probability density function at the point where the interclick time no longer represents the time between requests, but rather the time between entire sessions. Instead, it turns out that the distributions of interclick times for each of the selected users could be closely approximated by power-law distributions $\Pr(\Delta t) \sim \Delta t^{-\tau}$ over nearly six orders of magnitude, as shown in Figure 4.3. Moreover, in each case, it appears that $tau < 2$, suggesting that there is no central tendency at all to the time between a user's requests and that no delay can really be considered atypical of any of these users.

The possibility remained that these randomly selected user might be outliers, so I constructed an automated process for calculating the probability density function for a user's distribution of

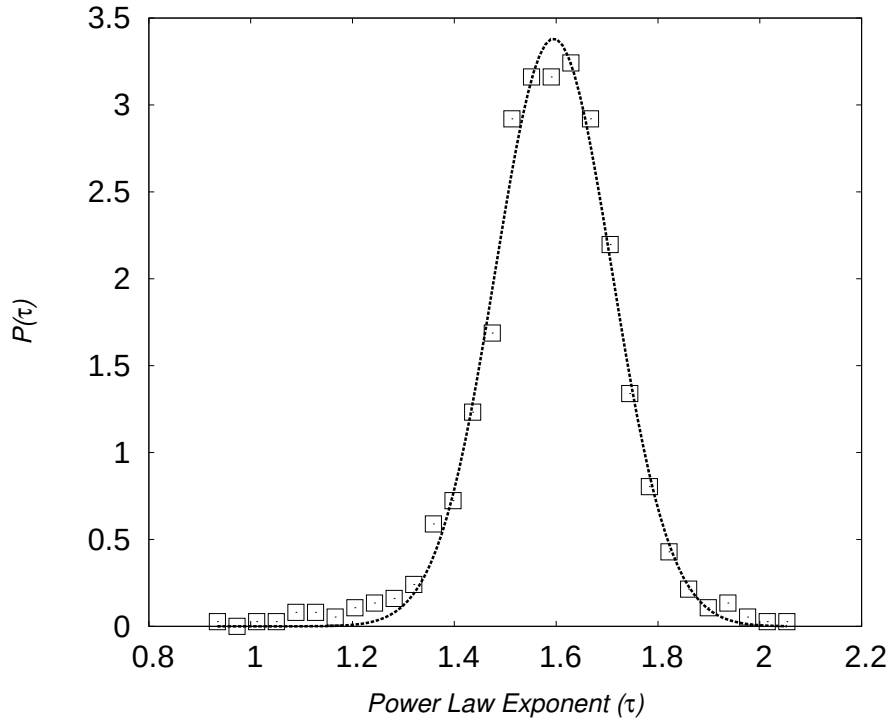


Figure 4.27: The distribution of the exponent τ for the best power-law approximation to the distribution of interclick times for each user. The fit is a normal distribution with mean $\langle\tau\rangle = 1.6$ and standard deviation $\sigma = 0.1$. These low values of τ indicate unbounded variance and the lack of any central tendency in the time between requests for a typical Web surfer.

interclick times using log-binned histograms and then fitting the results to a power law approximation. In the interest of computing time, the fitting procedure was more lax (least squares rather than maximum likelihood), but the process was still able to fit each distribution to a power law with a mean R^2 value of 0.989. The resulting distribution of power-law exponents, shown in Figure 4.3, is strongly normal, with a mean value $\langle\tau\rangle \approx 1.6$. This confirmed the finding that interclick times have no central tendency; in fact, scale-free behavior is so pervasive throughout the user population that a user agent exhibiting *regularity* in the timing of its requests would constitute an anomaly. These results make it clear that a robust definition of Web session cannot be based on a simple timeout.

The next natural approach might be to segment a click stream into sessions based on the rolling average of the number of clicks per unit time dropping below a threshold value. This approach turns out to be even more problematic than using a simple timeout, as there are now two parameters to consider: the width of the window for the rolling average and the threshold value. IF the window selected is too narrow, then the rolling average will often drop all the way to zero, and the scheme becomes prey to the same problems as the simple timeout. If the window selected is too large, then the rolling average is so insensitive to change that meaningful segmentation is impossible. In the end, the choice is once again made arbitrary, and we fail to gain any real understanding of user behavior. Moreover, examination of the moving average click rate for several users shows that the magnitudes of the spikes in the click rate fit a smooth distribution. This makes the selection of a threshold value just as arbitrary as it was before: the number of sessions becomes a function of the threshold rather than a feature of the data itself.

However, a third path does exist. The previous methods discussed identify each request as a single data point based on its target URL, but we have already seen the utility in treating requests as clicks: that is, as directed edge. Logging both the referring URL and target URL for each click makes possible another, more robust approach to constructing user session. We can expand on the notion of *referrer trees* as described in [130] to segment a user's click stream into a set of *logical sessions* based on the following algorithm:

1. Initialize the set of sessions T and the map $N : U \mapsto T$ from URLs to sessions.
2. For each request with referrer r and target u:
   (a) If r is the empty referrer, create a new session t with root node u, and set $N(u) = t$.
   (b) Otherwise, if the session $N(r)$ is well-defined, attach u to $N(r)$ if necessary, and set $N(u) = N(r)$.

(c) Otherwise, create a new session t with root node r and single leaf node u, and set $N(r) = N(u) = t$.

This algorithm assembles clicks into sessions based on the referring URL of a click matching the target URL of a previous click. Clicks are assigned to the session with the most recent use of the referring URL. Furthermore, each instance of a request with an empty referrer generates a new logical session.

Before taking a look at the properties of the logical sessions defined by this algorithm, we must highlight the differences between these logical sessions and our existing intuitive notion of a Web session. A logical session does *not* represent a period of time in which a user opens a Web browser, visits some set of Web sites, and then leaves the computer. It instead connects requests related to the same browsing behavior, grouping clicks based on commonality of purpose rather than proximity in time. If the user opens links in multiple tabs or uses the browser's back button, the subsequent requests will all be part of the same logical session. If the user then jumps directly to a search engine, they will start a new logical session; they have not left their computer, but they are beginning a new cognitive process. Tabbed browsing and the use of the back button make it entirely possible for a user to have multiple active logical sessions at any point in time. A user who always keeps a popular news site open in a browser tab might have a logical session related to that site that lasts for entire collection period. Logical sessions thus segment a user's click stream into *tasks* rather than time intervals. They also enjoy the advantage of being defined without reference to any external parameters, making their properties comparable across data sets and insensitive to the judgments of individual researchers.

The first statistics of interest for these logical sessions concern their tree structures. The number of nodes in the tree is a count of the number of requests in the session. Figure 4.3A shows the probability density function for the per-user distribution of the average size of a logical session.

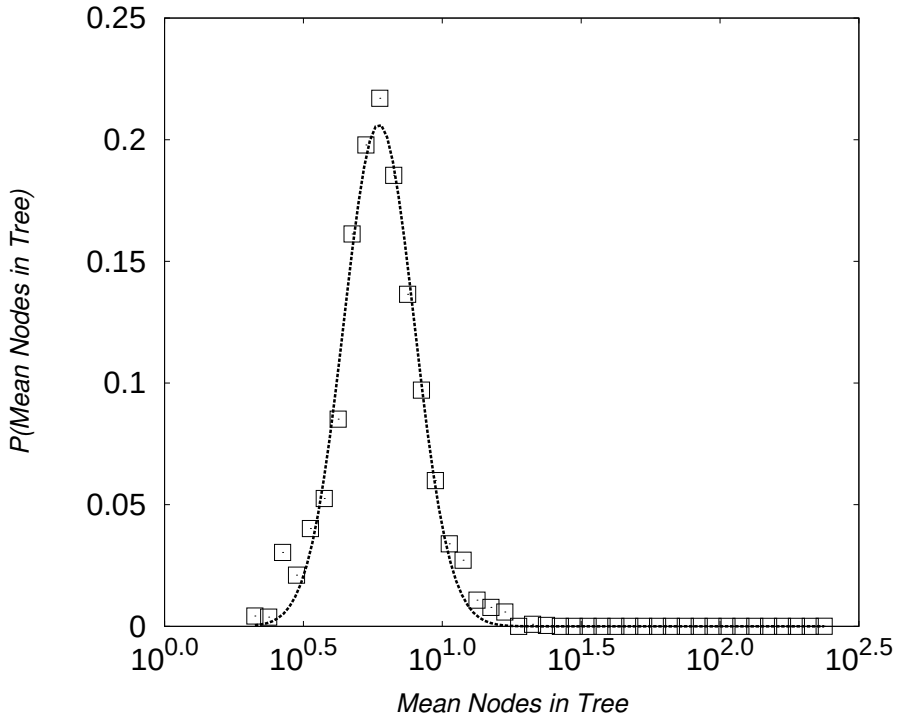
This distribution is well-approximated with a log-normal function, showing that the typical user has a mean of around 6.1 requests per session.

Along similar lines, the depth of the trees indicates the extent to which users follow chains of links during a Web-related task. Users who prefer surfing to searching would thus be expected to have deeper session trees. In Figure 4.3B, we can see the distribution of the average depth of the logical sessions for each user. It is again well-approximated with a log-normal, showing that a typical user's sessions have a depth of about three links. In other words, a typical session usually travels no more than two clicks away from the page on which it began.

The ratio between the number of nodes in each tree and its depth is of great interest. If this ratio is equal to 1, then the tree is just a sequence of clicks without any branching or backtracking, which corresponds to the assumptions of the random walker model. As this ratio grows past 1, the branching factor of the tree increases, the assumptions of PageRank break down, and a random walker must either backtrack or split into multiple agents. Figure 4.3 shows the distribution of the average node-to-depth ratio for the logical sessions of each user, which is well-approximated by a normal distribution with mean $\langle \mu \rangle = 1.94$. We thus see that sessions have structure that cannot be accurately modeled by a stateless random walker: there must be some provision for the backtracking and branching observed in the empirical data.

Although there is a strong central tendency for the mean size and depth of the logical sessions associated with each user, the same does not hold for logical sessions in general, i.e., when we consider all logical sessions as members of a single distribution. Figure 4.3 shows that distributions of the node count and depth for logical sessions aggregated across all users. When we remove the per-user identifying information in this way, we are once again confronted with heavy-tailed distributions exhibiting unbounded variance. This implies that the detection of automated browsing traffic is a much more tractable task if some form of client identity can be retained.

A



B

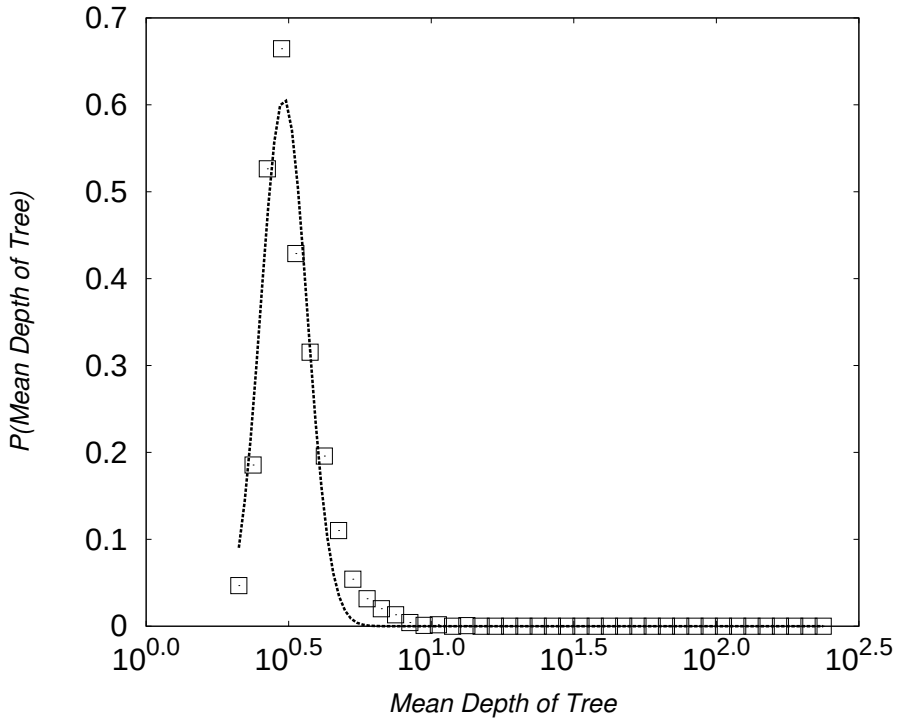


Figure 4.28: Distributions of the mean number of requests per logical session per user (A) and the mean depth of logical sessions per user (B). In both cases, we consider only non-trivial trees. Also included are reference log-normal fits for each distribution.

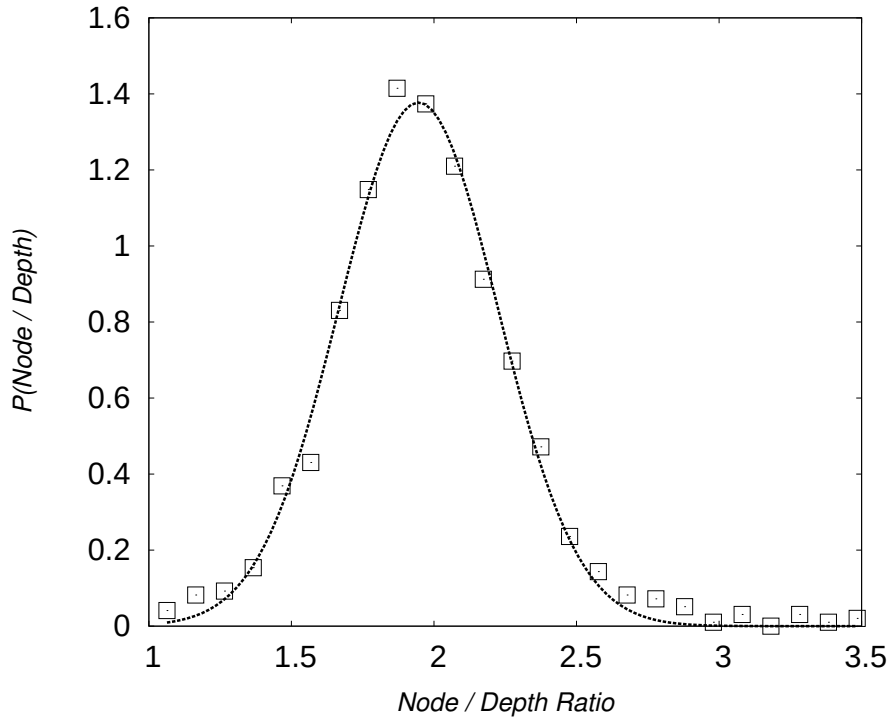
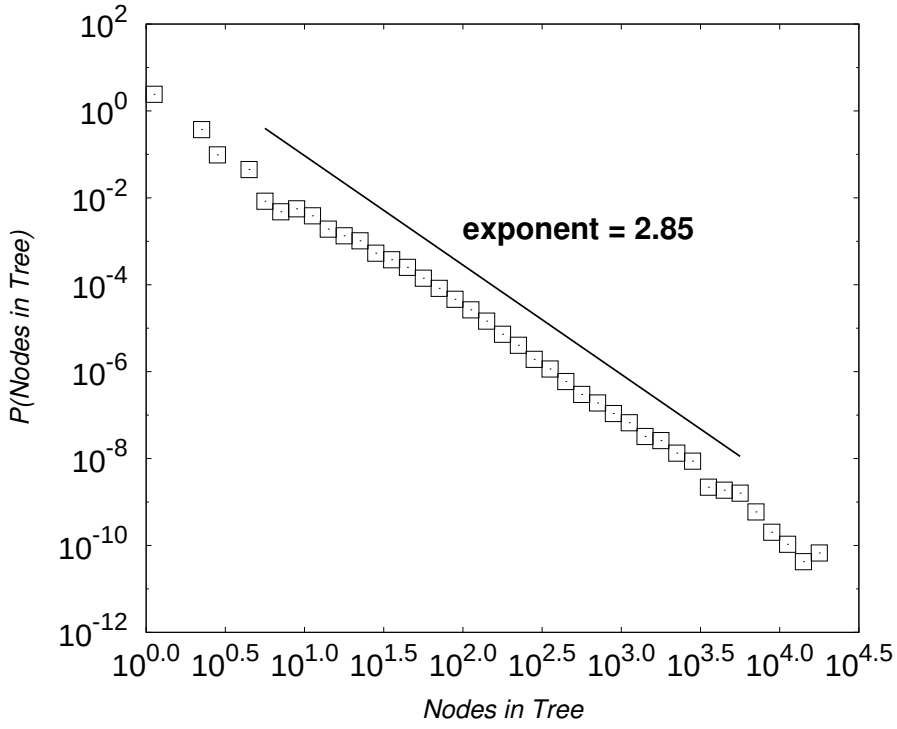


Figure 4.29: The distribution of the average ratio of the node count to the tree depth for the logical sessions of each user. The fit is a normal distribution with mean $\mu = 1.94$ and standard deviation $\sigma = 0.28$, showing that the branching factor of logical sessions is significantly greater than one.

A



B

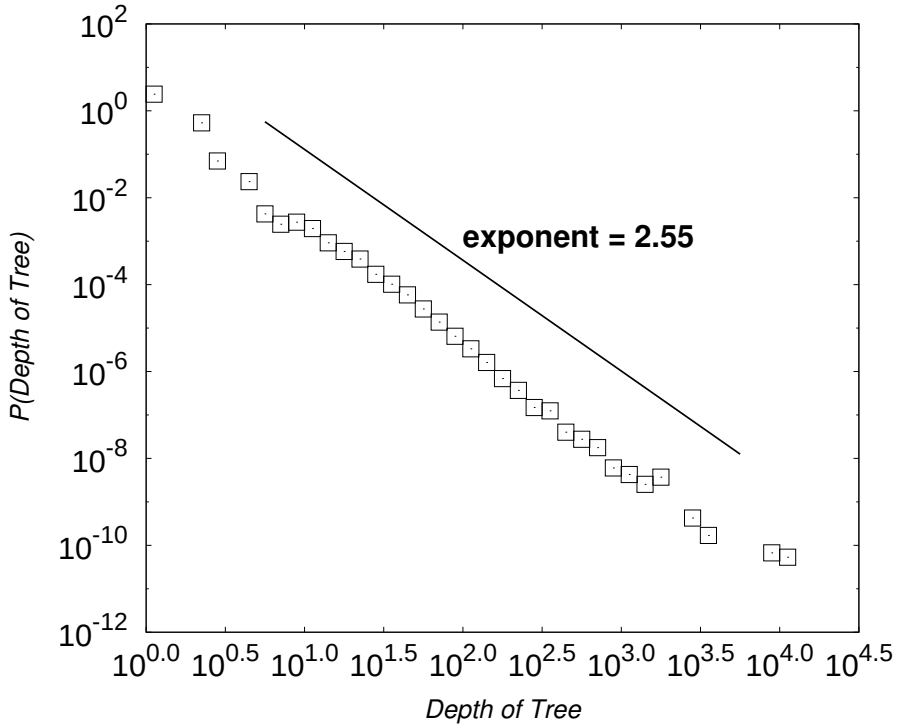


Figure 4.30: Overall distributions of the number of requests per logical session (A) and the depth of each logical session (B), with reference power-law fits.

Even though we have defined sessions logically, they can still be considered from the perspective of time. If we calculate the difference between the timestamp of the request that first created the session and the timestamp of the most recent request that added a leaf node, we obtain the duration of the logical session. When we examine the distribution of the durations of the sessions of a user, we encounter the same situation as we had in the case of interclick times: power-law distributions $\Pr(\Delta t) \sim \Delta t^{\tau}$ for every user. Furthermore, when we proceed along similar lines as before and consider the exponent of the best power-law fit for each user's data, we find the values are normally distributed with a mean value $\langle \tau \rangle \approx 1.2$, as shown in Figure 4.3. No user has a well-defined mean duration for their logical sessions. As had been suggested by the statistics of interclick times, the presence of strong regularity in a user's session behavior would be anomalous.

It is natural to speculate that we can get the best of both worlds by extending the definition of a logical session to include a timeout as an additional constraint, similar to the method employed in previous work on referrer trees [130]. Such a change is quite straightforward to implement: we simply modify the algorithm so that a request cannot attach to an existing session unless the attachment point was itself added within the timeout. This allow us to have one branch of the browsing tree time out while still allowing attachment on a more active branch.

While the idea is reasonable, it must be applied with great caution. We find that the addition of such a timeout mechanism once again makes the statistics of the sessions strongly dependent on the particular timeout selected—to a point. As shown in Figure 4.3, the number of sessions per user, mean node count, mean depth, and ratio of nodes to tree depth are all initially extremely dependent on the timeout. However, in contrast to session defined purely by timeout, this dependence becomes smaller as the timeout increases, suggesting that logical sessions with a timeout of around 15 minutes may be a reasonable compromise for modeling and further analysis.

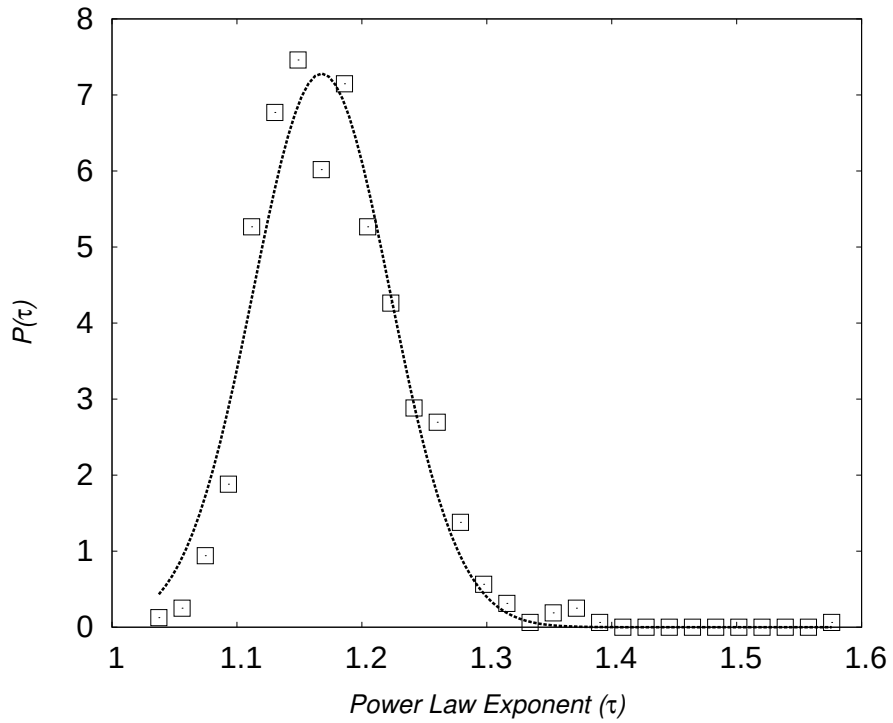


Figure 4.31: The distribution of the exponent τ for the best power-law approximation to the distribution of logical session duration for each user. The fit is a normal distribution with mean $\langle\mu\rangle 1.2$ and standard deviation $\sigma = 0.06$. These low values of τ indicate unbounded variance and the lack of any central tendency in the duration of a logical session.

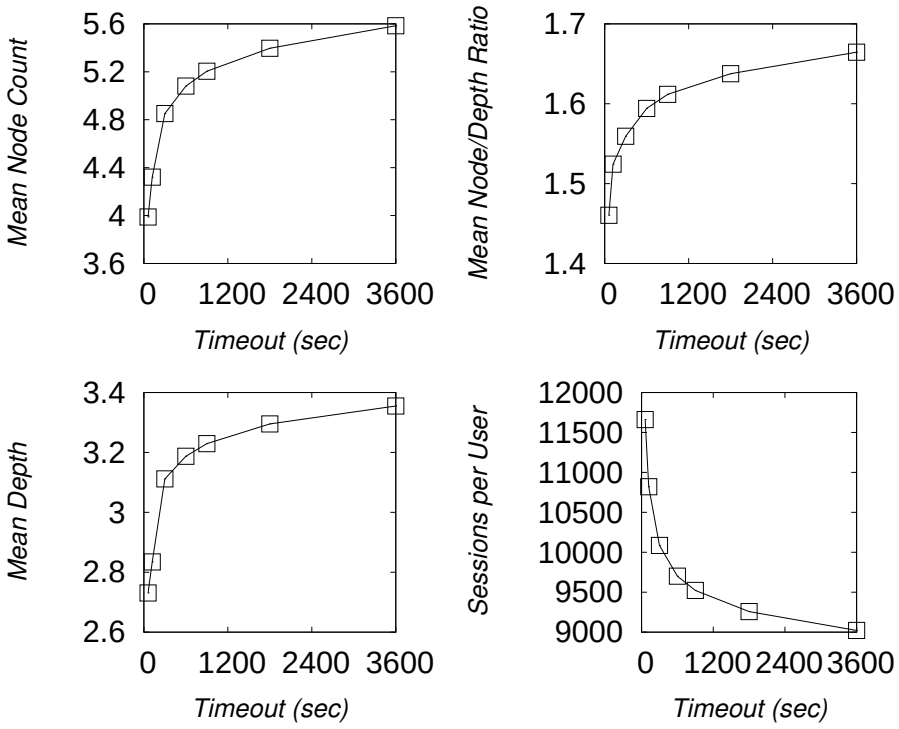


Figure 4.32: Logical session statistics as a function of timeout. Top left: Mean node count. Top right: Mean tree depth. Bottom left: Mean ratio of node count to tree depth. Bottom right: Mean number of sessions per user.

4.4 Traffic modeling

Armed with this extensive portfolio of information about the violation of the assumptions of the random surfer model, the time has finally come to attempt to construct a model that is better able to replicate features of actual traffic. Such a model must include only as much complexity as necessity and not only produce results similar to observed traffic, but also do so in a behaviorally plausible way.

Construction of this model will proceed in several steps. First I introduce the BookRank model, which extends PageRank by introducing state to the random surfer. I then present a more sophisticated model, ABC (Agents, Bookmarks, and Clicks), which combines BookRank with an agent-based generative model for sessions to yield a surprisingly robust and effective model of real Web traffic.

BookRank model

We have seen that a principal shortcoming of PageRank and other stateless models of Web navigation lies in their users' lack of memory. In the random surfer model, each user jumps from one page to another without any record of where they have been before or intend to go. Because they cannot purposefully return to a page, these random surfer cannot form the well-defined habits of navigation that are prevalent in empirical data and familiar to our own experience with the Web. The BookRank model described here introduces state to random surfers through the use of bookmarks: each agent maintains a list of bookmarks ranked by the number of previous visits to the page.

I must also pause to emphasize that credit for the development of the BookRank model belongs

much more strongly to my collaborators than to myself, Bruno Gonçalves in particular [72]. However, thorough exposition of this model is necessary to build a foundation for the ABC model, for which I took a much stronger role in development.

Note that the introduction of state to the agent does entail a major sacrifice: while the PageRank vector can be calculated in an efficient and straightforward way using the power method (as well as a variety of other techniques that accelerate the process), introducing this sort of diversity among the random surfers precludes any such "all at once" calculation. The results presented here thus rely on repeated simulation of individual agents, either in serial or in parallel, and do not represent a steady-state distribution of visitation over the Web graph. The algorithm followed by each agent in the BookRank model is detailed below.

Initially, each agent randomly selects a starting site (node). Thereafter, at each time step:

1. Unless previously visited, the current node is added to the bookmark list. The number of times each node has been visited is recorded, and the list of bookmarks is kept ranked from the most to least visited.

2. With probability p_t, the agent teleports (i.e., jumps) to a previously visited site drawn from the bookmark list. The bookmark with rank R is chosen with probability $P(R) \propto R^{-\beta}$.

3. Otherwise, with probability $1 - p_t$, the agent navigates locally by following a link from the present node. There are two alternatives:

   (a) With probability p_b, the agent simulates the use of the back button by returning to the previous node.

   (b) Otherwise, with probability $1-p_b$, an outgoing link is clicked, with a uniform probability among the possible choices.

Note that this definition of BookRank is actually a generalization of PageRank. If we set $\beta = p_b = 0$, we recover the original PageRank model.

The performance of BookRank is now evaluated through an attempt to use it to recover key aspects of the empirical data from the dormitory study. The parameters p_t and β are held constant, and the model is then tested using several values of p_b. The values of the fixed parameters are drawn from real-world experience. We set $p_t = 0.15$ to correspond with the typical teleportation parameter α used with PageRank. The value of β based on the distribution of empty-referrer traffic in the empirical data. This follows from work by Fortunato *et al.* that demonstrates that the ranking exponent β and power-law exponent of the empty-referrer distribution α should satisfy the relationship $\beta = 1/(\alpha - 1)$ [66]. We find that $\alpha \approx 1.7$ for the empirical data, leading to the choice $\beta = 1.4$.

Of course, the simulations require an actual network substrate on which to run the agents. This evaluation employs artificial scale-free networks with N nodes and degree distribution $P(k) \sim k^{-\gamma}$ generated using a growth model from the literature [66]. We set $N = 6.3 \times 10^5$ and $\gamma = 2.1$ to match the subset of the Web graph sampled in the dormitory study.

The actual simulations of the PageRank and BookRank models were repeated for a series of 250 different generated networks. For each network, we simulated the activity of 1000 agents with 10^5 clicks allocated to each, again broadly matching the empirical data. (Note that in the empirical data, the number of requests per user was a broad log-normal distribution, whereas for the purpose of this evaluation we assume it to be constant.)

Let us first consider the aggregate distribution of traffic received by sites. In the earlier studies in this chapter, this distribution was referred to as in-strength ($s_i n$); in this discussion of modeling, I shift to the simpler notation T for traffic, since we are no longer particularly concerned with

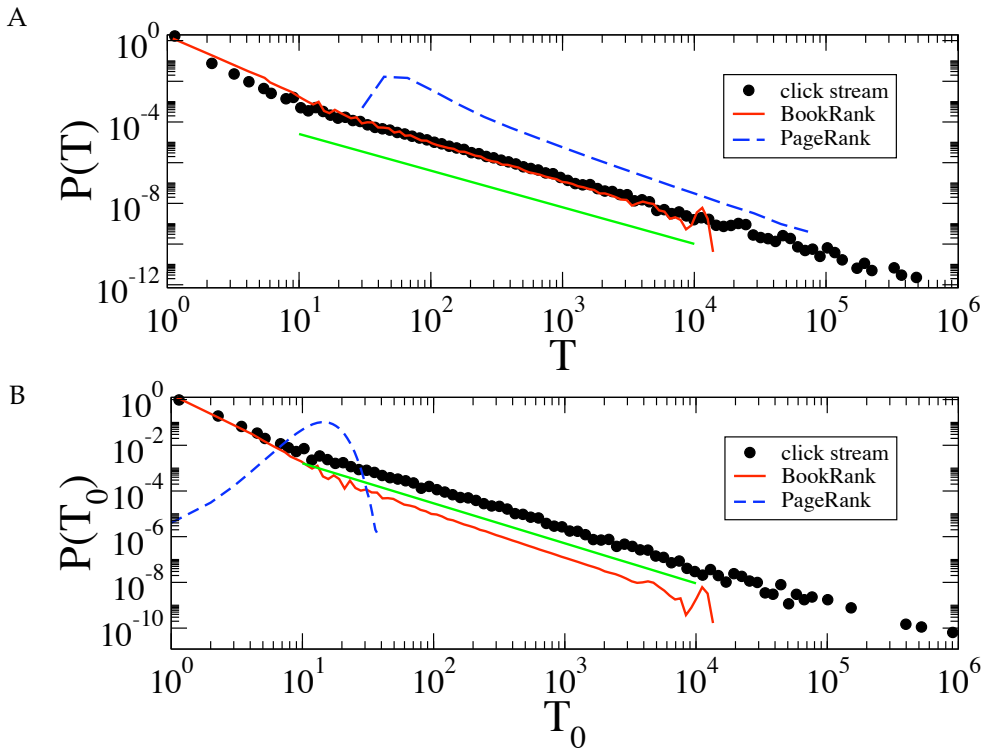


Figure 4.33: Distributions of all traffic to sites T (A) and traffic originating only from jumps to sites T_0 (B) for the empirical click stream, PageRank, and BookRank.

out-strength. (Similarly, s_0 will become T_0 for the remainder of this analysis.) The traffic distribution $P(T)$ is shown in Figure 4.4A for PageRank, BookRank, and the empirical data. All three distributions are well-approximated by power laws of the form $P(T) \sim T^{-\eta}$.

In the case of PageRank, we recover an exponent $\eta \approx 2.1$, which matches that of the degree distribution in the Web graph. This result is well-known in the literature [30] and happens simply because random walkers visit the nodes of an undirected network with a probability proportional to their degree. The exponent changes very little when we introduce directed links and a small teleportation probability, as used in PageRank. This exponent corresponds poorly to that of the empirical distribution, which yields the estimate $\eta \approx 1.75$, as shown by the visual guide in Figure 4.4A. This discrepancy, together with the robustness of random walk exponents, indicates that

the empirical data indeed reflect some other mechanism beyond PageRank's choice of the next destination. The BookRank model, on the other hand, provides an excellent prediction of the empirical traffic distribution.

In Figure 4.4B, we see the distribution of traffic T_0 originating from jumps, identified in the empirical data by HTTP requests with an empty referrer. Once again, the prediction of PageRank is poor, since it assume a uniform probability $1/N$ of restarting from any node. In contrast, BookRank selects the next site using the bookmarks of each agent; its prediction and the empirical data both display power-law behavior with exponent $\alpha \approx 1.7$. This is consistent with the selection of the fixed parameter $\beta = 1.4$ and serves as a strong indication that the dependence of BookRank on the rank of bookmarks for choosing teleportation targets captures real user behavior in a quantitative way. Moreover, the exponent $\beta \approx 1.4$ is quite similar to the one empirically observed in the distribution of click probability as a function of the ranks of results returned by a search engine [41, 65]. This provides further support for the hypothesis that rank-based choice is a key mechanism underlying Web search and browsing behavior.

A second distribution of interest is the distribution of traffic across links $P(\omega)$, shown in Figure 4.4. PageRank predicts a log-normal curve with well-defined mean and variance. The empirical data, in contrast, reveal a much wider power-law distribution $P(\omega) \sim \omega^{-\delta}$ with $\delta \approx 1.9$. BookRank provides a much better approximation to the empirical statistics than does PageRank, though the distribution is somewhat more steep and narrow.

A final quantity of interest in measuring the spread of activity of a user or agent through the network is to sample the time required to return to a page after a previous visit. Figure 4.4 illustrates the distribution of return times τ as measured as a number of clicks. Once again, the prediction generated by BookRank is significantly closer to the empirical distribution $P(\tau)$ than PageRank, which produces a distribution with a smaller power-law exponent, corresponding to an

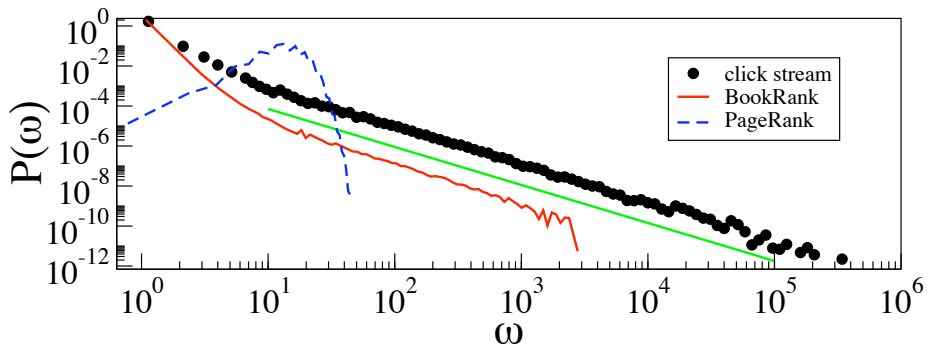


Figure 4.34: Distributions of link traffic ω for the empirical click stream, PageRank, and BookRank. PageRank predicts a far narrower distribution than the broad power-law distributions exhibited by the empirical data and BookRank.

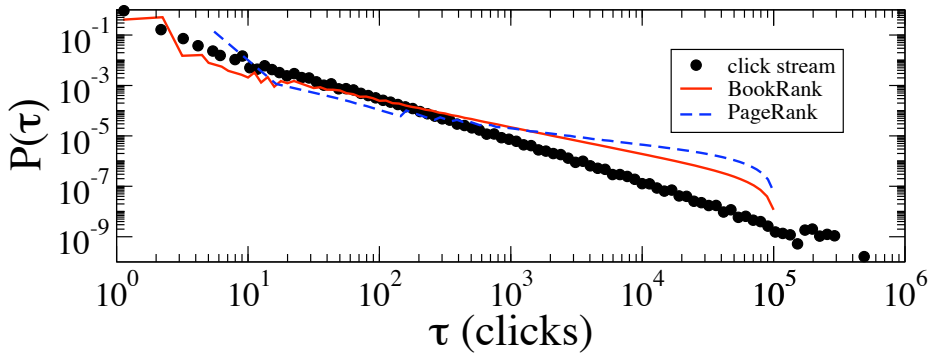


Figure 4.35: Distributions of time τ between consecutive visits to the same site for the empirical click stream, PageRank, and BookRank. Both PageRank and BookRank overestimate return times, but the performance of BookRank more closely resembles the empirical distribution.

overestimate of the time required for a user to return to a site. This is a basic consequence of the lack of internal state for random walkers: they have no notion of where they have been and have no tendency to return beyond that dictated by the topology of the network. The BookRank model improves on this performance by introducing memory in the form of its bookmark list.

This model constitutes a clear step in the right direction but shares some of the limitations present in previous traffic models. The probability of ending a session is still taken to be uniform, resulting in an exponential distribution of session lengths, while the empirical data obey a

far broader distribution, as we saw in the previous section. Furthermore, the discrepancy between entropy for the empirical data and BookRank is quite significant, suggesting that BookRank's implementation of the back button and teleportation probabilities still leaves something to be desired. A better model of individual sessions is needed that provides a better mechanism for ending sessions than an uniform teleportation probability.

ABC model

We are finally ready to examine the culmination of this study of Web traffic: the ABC (Agents, Bookmarks, and Clicks) model, which combines elements of the BookRank model with an agent-based generative model for individual sessions. As we shall see, this model proves exceedingly capable of reproducing a variety of distributions from empirical data while remaining behaviorally plausible.

The empirical data involved in this final study are once again drawn from the dormitory study. The logical sessions formed from the click streams of individual users are coupled with a half-hour timeout introduced for the sake of behavioral plausibility, meaning that a click cannot not be associated with a session tree that has not received any additional requests within the past thirty minutes. This has little overall effect; recall from the earlier results that the timeout dependence of logical sessions is largely absent for values of greater than fifteen minutes.

The reasons for attempting to integrate logical sessions into a model of Web traffic are straightforward. They reflect the multitasking behavior of users in the age of tabbed browsing. They also allow us to infer how users backtrack as they browse. Because modern browsers follow sophisticated caching mechanisms to improve performance, unless overridden by HTTP options, a browser will generally not issue another request for a recently accessed page. This causes many page revisits to be absent from the click stream and prevents us from observing multiple links pointing to the

same page within a single logical session. While logical sessions give us no *direct* way of determining when a user presses the back button, they do allow us to *infer* information about backwards traffic: if the next request in the session comes from a URL other than the most recently visited one, the user must have navigated to that page or opened it in a separate tab. While we can reasonably expect these behaviors to reflect many properties of empirical traffic, they are entirely absent from the standard random surfer model and only partly accounted for by the p_b parameter of BookRank.

This study emphasizes a more formal comparison of the ABC model to the empirical data and the predictions of simpler models. In particular, six properties of traffic data are selected as a standard of comparison. All of these have been introduced earlier in this chapter, but are summarized here again to make the goals of this final study more clear.

The six properties we seek to model are as follows:

Page traffic The total number of visits to each page, denoted as T. Bear in mind that because of caching mechanisms, the majority of revisits to a page by a single user beyond the first visit within a logical session will not be represented in the data.

Link traffic Total total number of times each link between pages has been traversed by a user, denoted as ω. Links are identified by the referrer and destination URLs in each request. Again, recall that caching behavior causes us to typically observe only the first click to a destination page within each session.

Empty referrer traffic The number of times each page is used to initiate a new session, denoted as T_0. We assume that a request without a referring page corresponds to the user beginning a new Web-related task by using a bookmark, opening a link from another application, or manually entering an address.

Entropy Shannon information entropy of a user's traffic distribution, denoted as S. For an individual user j, we define the entropy as $S_j = \sum_i \rho_{ij} \log_2 \rho_{ij}$, where ρ_{ij} is the fraction of visits to site i for user j, aggregated across sessions. Given n visits to a collection of sites, the entropy will be at its minimum $S = 0$ when all visits are to a single site, while its maximum $S = \log_2 n$ is achieved when a single visit has been paid to each of n distinct sites. Entropy can offer a better view into the behavior of a single user than the number of unique sites visited, since two users who have visited the same number of sites can have very different measures of entropy. We can loosely interpret the entropy as a measure of the information needed to describe the browsing pattern of a single user or agent.

Session size The number of unique pages visited in a logical session, denoted as N_s.

Session depth The maximum tree distance between the starting page of a session and any page visited within the same session, denoted as N_d. Recall that logical sessions have a tree-like structure, since requests that go back to a previously visited page are generally served from the browser cache.

The behavior of the empirical data with respect to these properties has already been presented at the host level; in this final study, we zoom in further and focus on individual pages rather than sites. The revised dimensions of the data set caused by this increased detail are summarized in Table 4.4. Note that this study omits the distribution of return time (i.e., the number of clicks between two consecutive visits to the same page by a given user) presented in the discussion of BookRank because of the segmentation of the data into logical session. The existence of overlapping sessions, coupled with caching behavior, mean that this information cannot be retrieved in a reliable way from the empirical data for the purpose at hand.

Each of these distributions are compared with two baseline reference models. The first reference

Table 4.4: Dimensions of the page-level view of the dormitory data set.

Page requests	29,494,409
Unique users	967
Unique URLs	2,503,002
Unique target URLs	2,084,031
Unique source URLs	864,420
Number of sessions	11,174,254
Mean sessions per user	11,556

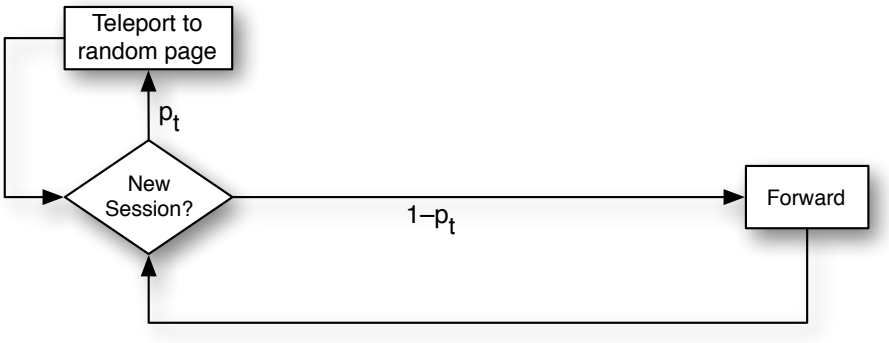


Figure 4.36: Schematic illustration of the agent-based PageRank model.

model is referred to simply as PageRank for the sake of brevity, but it does not represent the customary steady-state distribution of PageRank, since this would provide information only about T and ω. Instead, we imagine a population of PageRank random walkers, as many as the users in the dormitory study, each using the same teleportation probability $p_t = 0.15$. Each one of these walkers browses for as many sessions as there were empirical sessions for the corresponding real-world user, as depicted in Figure 4.4. The PageRank sessions are terminated by the constant-probability jumps, so the total number of pages visited by a walker may be quite different from the number visited by the user. Consistent with the traditional formulation of PageRank, teleportation jumps lead to session-starting pages selected uniformly at random.

The second reference model is BookRank, with the probability of backward navigation p_b set

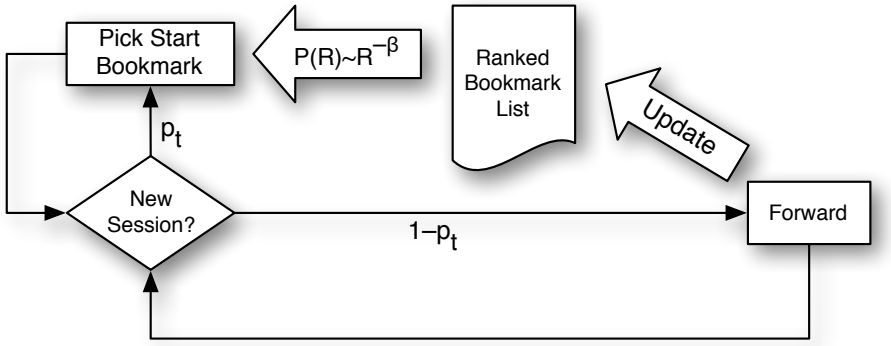


Figure 4.37: Schematic illustration of the simplified BookRank model.

to zero, which eliminates the simulation of the back button that complicated the previous results. This version of BookRank is thus distinguished from PageRank solely through its use of memory; the BookRank agents still maintain individual lists of bookmarks that are chosen as teleportation targets based on the number of previous visits. The BookRank algorithm with the $p_b = 0$ simplification is shown in Figure 4.4. As in the case of the first reference model, the BookRank agents browse for as many sessions as there were for the corresponding real-world user.

Once again, the reference models are simulated on artificial scale-free network with N nodes and degree distribution $P(k) \sim k^{-\gamma}$ generated using the growth model of Fortunato *et al.* [66]. The degree distribution is still held at $\gamma = 2.1$ to mimic the actual Web graph, and the size of the artificial network is increased to $N = 10^7$ to ensure that the artificial network is substantially larger than the number of pages visited in the empirical data. This graph is constructed with symmetric links to prevent agents from being stranded in dangling links; as a result, the node of each in-degree is equal to its out-degree.

Within each session for one of the reference models, we simulate the operation of the browser cache by recording traffic for links and pages only for the first visit to the target page in a particular session. This makes it possible to measure the number of unique pages visited in a session for the

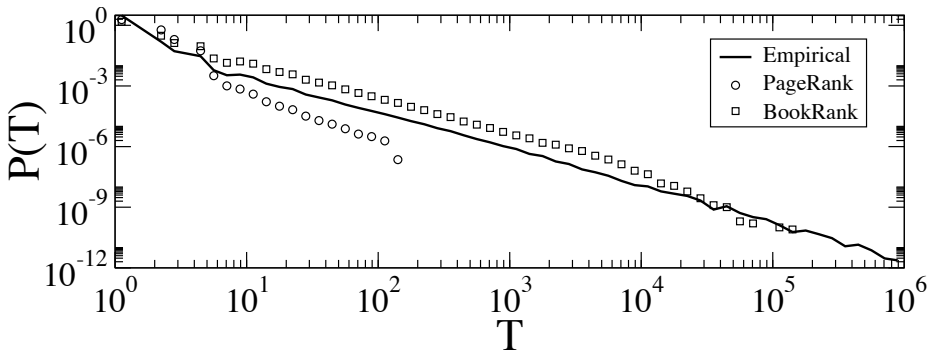


Figure 4.38: Distributions of page traffic T for the empirical data and baseline models.

reference models, which is directly comparable to the logical sessions from the empirical data. We assume that these cached pages are reset between sessions.

The aggregate distributions of traffic to individual pages obtained by the reference models are compared to the empirical data in Figure 4.4. The empirical data show a familiar broad power-law distribution for page traffic $P(T) \sim T^{-\alpha}$ with exponent $\alpha \approx 1.75$, matching the results of the host-level analysis.

Theoretical arguments from the literature suggest that PageRank ought to behave in a similar fashion [123]. If we disregard the teleportation mechanism (which is simply a session delimiter for this simulation), a node with in-degree k can anticipate a visit whenever one of its neighbors has been visited in the previous time step. The traffic it receives will therefore be proportional to its degree if no degree-degree correlations are present in the graph. This intuition and the previous results in this chapter combine to predict that PageRank should produce a page-level traffic distribution described by a power law $P(T) \sim T^{-\alpha}$, where $\alpha \approx 2.1$ matches the distribution of the in-degree [30, 64].

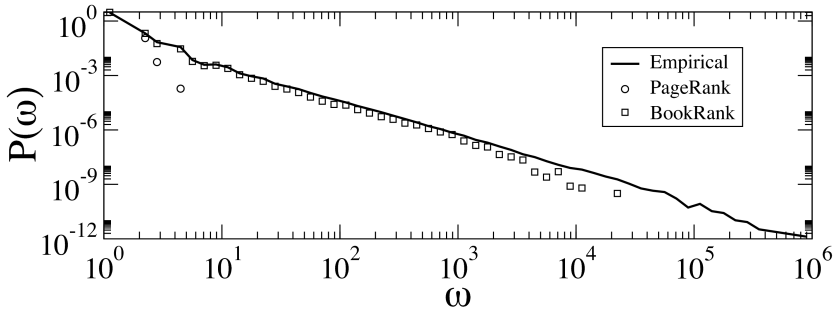


Figure 4.39: Distributions of link traffic ω for the empirical data and baseline models.

The PageRank distribution seen in Figure 4.4 is indeed consistent with this prediction. The traffic generated by BookRank, in contrast, is strongly biased toward previously visited pages (bookmarks). It therefore shows a distribution that is three orders of magnitude more broad, in better agreement with the empirical data.

The distribution of weights ω across the links between pages reflects the diversity of traffic crossing each hyperlink in the Web graph. Figure 4.4 compares the empirical distribution of link weights with those resulting from the baseline models. The data again reflect a very wide power-law approximation of $P(\omega)$ with an exponent of roughly 1.9, which is consistent with the host-level results and reinforces the finding that link weights do not have a well-defined mean.

The predictive performance of the PageRank and BookRank model as reflected in Figure 4.4 is a vivid illustration of the tremendous diversity of links when we consider the chances of their

actually being clicked. An informal argument can help to make sense of the poor performance of the PageRank reference model in reproducing the data. If we again disregard teleportation, the traffic received by a page is roughly proportional to the in-degree of that page. The traffic expected on a link would thus be *proportional* to the traffic into the page and *inversely proportional* to the out-degree of the page if we assume that links are chosen uniformly at random. Since the in-degree and out-degree of each node are equal in the artificial network, this leads to link traffic that is independent of the degree and thus more or less constant for every link. This is reflected in the quickly decaying distribution of link traffic for PageRank. In the case of BookRank, on the other hand, we have much stronger heterogeneity in the probability of visiting pages, which indirectly manifests itself as heterogeneity among link weights, resulting in a broad distribution that fits the empirical data much more closely, as shown in Figure 4.4.

The distributions of starting points or empty-referrer traffic T_0 are shown in Figure 4.4. The empirical data clearly indicate that all pages are not equally likely to be chosen as the starting point of a session. Indeed, their popularity as starting points is roughly distributed as a power law with an exponent of approximately 1.75, once again fairly consistent with the host-level results and implying a diverging variance and mean as the number of sessions in the data set increases. While this is not unexpected given the prior results and our intuition about human behavior, it does demonstrate how off the mark is the uniform teleportation assumption of PageRank's random walker model. The bookmarking mechanism, on the other hand, continues to perform well in capturing the non-uniform probability of starting pages. (Recall that the BookRank parameter β was chosen in the hopes of eliciting this behavior.)

We now turn from the aggregate properties of the system and attempt to characterize individual users. The simplest hypothesis would be that the broad distribution we have seen in aggregate user behavior (T, ω, and T_0) are simply a reflection of extreme variability within the traffic generated by

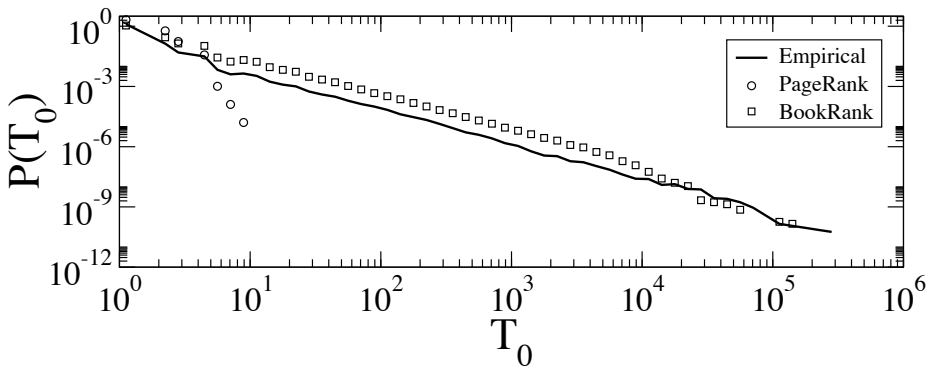


Figure 4.40: Distribution of empty-referrer traffic T_0 for the empirical data and baseline models.

individual users, leading to the conclusion that there is no such thing as a "typical" user in terms of generated traffic. We have already seen from the host-level analysis of the data that there is strong reason to doubt this hypothesis. To evaluate how diverse is the behavior within a group of users, we again employ Shannon's information entropy. Recall that given an arbitrary number of clicks n, the entropy is at its maximum ($S = \log_2 n$) when each of n pages are visited once, and at its minimum ($S = 0$) when all visits have been paid to a single page.

The distribution of entropy across users and agents is shown in Figure 4.4. We note first that the PageRank reference model produces much higher entropy than observed in the empirical data. This can be interpreted as a consequence of the fact that a PageRank walker picks starting pages with uniform probability, causing traffic to spread out from randomly selected points all over the graph. A real user is more likely to start from a previously visited page and therefore also more likely to revisit its neighboring pages. BookRank strongly emphasizes this repetitive behavior, and we do indeed observe lower entropy values for BookRank. However, BookRank goes too far in underestimating entropy and thus fails to capture much of the variability in the behavior of individual users.

Another note of explanation of these results is in order. The initial evaluation of BookRank

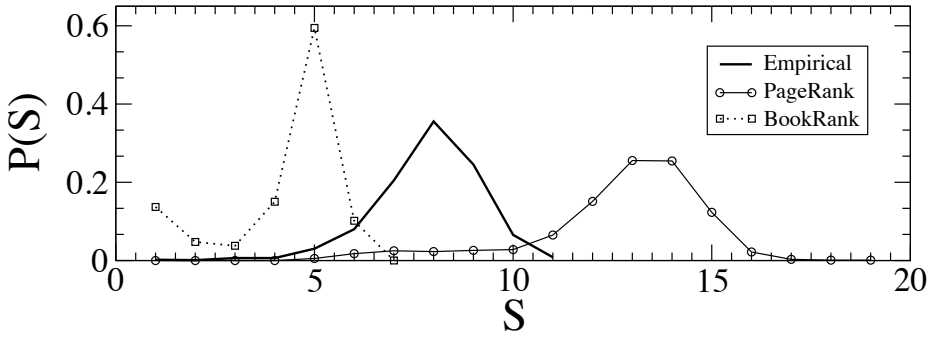


Figure 4.41: Distribution of Shannon entropy S for the empirical data and baseline models.

did include analysis of entropy (not reported here), finding that the entropy values produced by the model were greater than the empirical distribution, with the distribution moving to the left as the parameter p_b decreased—and yet here BookRank seems to yield lower entropy even with $p_b = 0$. This is a consequence of the different simulation conditions between the two studies. In the initial study, each agent generated clicks using the BookRank model until a uniform tally had been reached, which proved to be a poor approach. In this study, each BookRank agent generates clicks until it has gone through as many entire sessions as the corresponding real-world user.

Finally, we can again consider the distributions that characterize logical session: the size (number of unique pages, N_s) and the depth (distance from the starting page, N_d). These distributions, shown in Figures 4.4 and 4.4, are quite broad, spanning three orders of magnitude and indicating a surprisingly large proportion of very long long sessions. In contrast, both the PageRank and BookRank reference models generated very short sessions. This behavior is to be expected: the probabilistic teleportation factor that determines when a PageRank or BookRank walker starts a new session is incapable of capturing broadly distributed session sizes. In fact, session size is bounded by the length ℓ (number of clicks) of a session, which exhibits a narrow, exponential distribution $P(\ell) \sim (1-p_t)^\ell$. Note these exponentially short sessions are not inconsistent with the high entropy of PageRank walkers seen in Figure 4.4; they simply show that this diversity is a result of

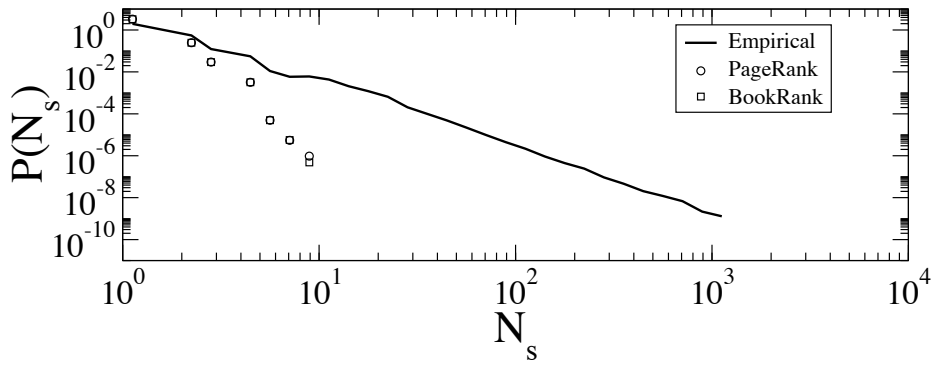


Figure 4.42: Distribution of session size N_s for the empirical data and baseline models.

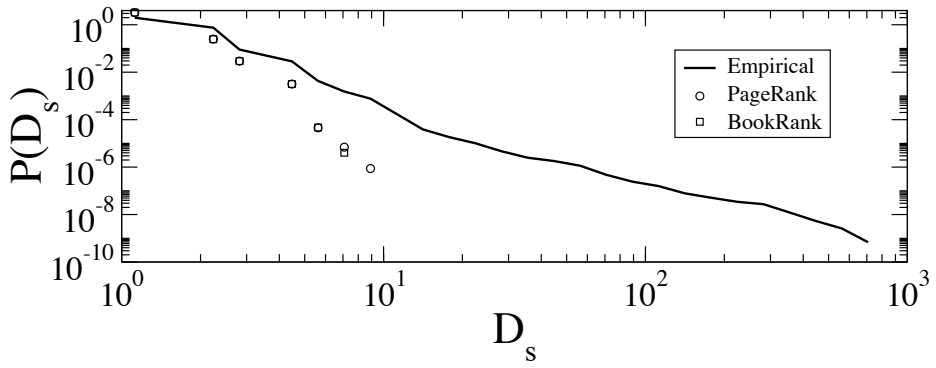


Figure 4.43: Distribution of session depth N_d for the empirical data and baseline models.

the frequent jumps to random targets rather than browsing behavior.

This analysis of the empirical data and baseline models demonstrates that a more sophisticated model of user behavior is needed to capture the observed user behavior as captured by the distributions T, ω, T_0, S, N_s, and N_d. We are finally ready to examine the proposed ABC model, which builds upon the BookRank model in several ways.

First, having verified through this analysis that a backtracking mechanism is indeed necessary to capture the tree-like structure of sessions, we restore the p_b parameter to the model. (See the top row of Figure 4.45 for examples.) This parameter was used in the initial formulation of BookRank to set the chance that an agent would return to the previously visited page rather than move forward

along a link. Recall the earlier finding that the incoming and outgoing traffic of a site are seldom equal, with the ratio between incoming and outgoing clicks being distributed over many orders of magnitude. This violation of flow conservation simply cannot be explained by teleportation alone, demonstrating that users' browsing session have definite branches. Recall also that the preceding results showed an average node-to-depth ratio for session trees of almost two. All of these observations are consistent with the widespread with use of tabs and the back button. Indeed, other studies have shown that users employ the back button quite frequently [44, 141]. We can therefore feel justifying in resuming the use of the back button to model any branching behavior.

The largest addition to the model has to do with the fact that the BookRank model fails to predict the statistics associated with individual users and sessions: all agents are identical, session size has a narrow, exponential distribution, and the comparison with the empirical entropy distribution is clearly unsatisfactory. In the real world, the duration of a session is related to its purpose. A session depends on the intentions, goals, and interests of a user, and different users have very different interests. Users visiting relevant pages, those whose topics match their interests, will lead to more clicks and therefore, longer sessions.

We attempt to capture this diversity by introducing two elements to the model: agents with distinct *interests* and page *topicality*. The basic notion is that an agent spends some of its limited store of attention when navigating to a new page, and attention is gained when the agent arrives at a page whose topics match the user's interests.

To model this process, we can imagine that each agent stores some real-valued "energy" (units of attention) while browsing. Visiting a new page incurs a higher energy cost than going back to a previously visited page. Pages already visited during a session yield no energy because of their lack of novelty, while unseen pages can increase the energy store by some random amount that depends on the page's relevance to the agent. Agents continue to browse until they run out

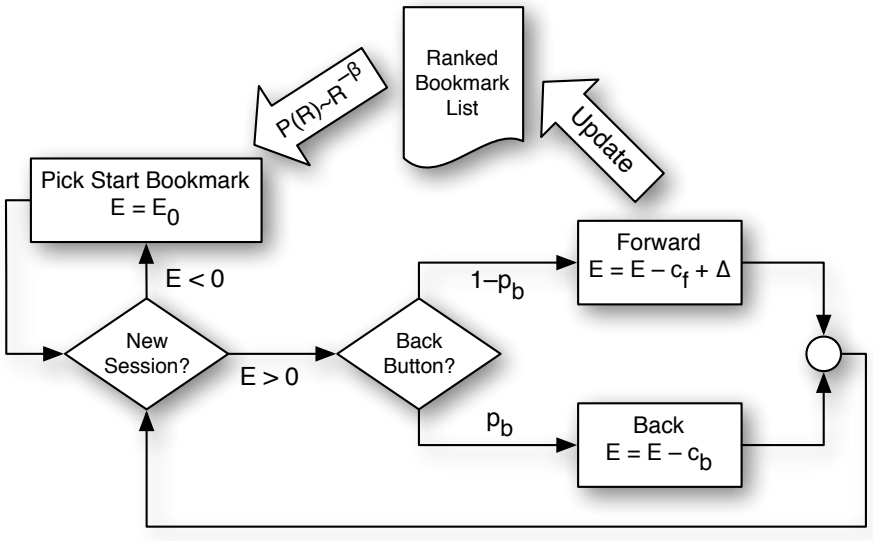


Figure 4.44: Schematic illustration of the ABC model.

of energy, whereupon the session is terminated and they start a new one in the same manner as BookRank.

The name of this combined model, *ABC*, is inspired by its main ingredients: agents, bookmarks, and clicks. The clicks it generates are driven by the topicality of pages and agent interests, a notion that is inspired in part by the *InfoSpiders* algorithms for topical crawlers [105, 110, 112]. These InfoSpiders were designed to explore the Web graph in an adaptive and intelligent fashion, driven by the similarity between search topics and page content. Better matches led to more energy and more exploration of local link neighborhood, while irrelevant pages caused agents to run out of energy and die, so that the system's limited resources could be allocated to more promising neighborhoods. This principle is now used in ABC to model browsing behavior.

A schematic overview of the ABC model is provided in Figure 4.4. The algorithm employed by each agent proceeds as follows. Each agent starts at a random page with an initial quantity of energy E_0. Then, for each time step:

1. If $E \leq 0$, the agent starts a new session (assuming the simulation is not complete) by teleporting to a bookmark, just as in BookRank.

2. Otherwise, if $E > 0$, the agent continues the current session, following a link from the present node. There are two alternatives:

   (a) With probability p_b, the agent uses the back button, which takes it back to the previous page (i.e., one step closer to the root of the session tree). The agent's energy is decreased by a fixed cost c_b.

   (b) Otherwise, with probability $1 - p_b$, a forward link is clicked, with uniform probability across all outgoing links. The agent's energy is updated to $E - c_f + \Delta$, where c_f is a fixed cost and Δ is a stochastic value representing the relevance of the new page to the agent. As in BookRank, the bookmark list is updated with new pages and kept ranked by visit frequency.

The dynamic variable Δ is a measure of the relevance of a page to a user's interests. The simplest way to model relevance is by a scalar random variable (e.g., drawn from a Gaussian distribution). In this case, the amount of stored energy regulates a random walking process. It has been shown that the session duration ℓ (number of clicks until the random walk reaches the termination condition $E = 0$) has a power-law tail $P(\ell) \sim \ell^{-\frac{3}{2}}$ [80]. However, the empirical results for session duration already presented suggest a larger exponent. More importantly, we know from empirical studies that the content similarity between two Web pages is correlated with their distance in the link graph, and so is the probability that a page is relevant with respect to some given topic [48, 109, 108]. Therefore, two neighboring pages are likely to be topically related, and the relevance of a page t to a user is related to the relevance of a page r that links to t.

To capture this *topical locality*, we can introduce correlations between the Δ values of consecutively visited pages. For the starting page in a session, we use an initial value $\Delta_0 = 1$. Then, when a page t is visited for the first time in that session, Δ_t is determined by

$$\Delta_t = \Delta_r(1 + \epsilon)$$

where r is the referrer page, ϵ is a random variable uniformly distributed in $[-\eta, \eta]$ and η is a parameter controlling the degree of topical locality. When the agent starts a new session, we assume that a page can again become interesting and thus provide the agent with energy, even if it was visited in a previous session. However, the same page will yield different quantities of energy in different sessions, based on changing user interests.

The evaluation of the ABC model involved two sets of simulations, in which the agents navigate two distinct scale-free graphs. The first graph (G1) is the artificial network already used for the evaluation of PageRank and BookRank. Recall that for this network, $N = 10^7$ nodes and the degree distribution obeys a power law with exponent $\gamma = 2.1$ to match the Web graph. The second graph (G2) is derived from an independent data set gathered from the same source as in Section 4.1. This graph is based on three weeks of traffic collected in November 2009 and is aggregated at the page level. We use only the largest strongly connected component from the undirected graph; this removes the danger of dangling links and becoming stuck in small, isolated components. The nodes correspond to actual visited pages and the edges to actual traversed links. The component contains $N = 8.14 \times 10^6$ nodes and the same degree distribution, with exponent $\gamma \approx 2.1$.

Within each session we simulate the operation of the browser's cache as discussed earlier, so that the measure of unique pages visited by the model agents will be directly comparable with the empirical session size.

Running the simulation requires setting several model parameters. Recall from the initial presentation of BookRank that we can expect the bookmark selection parameter β to obey the relation $\beta = 1/(\alpha - 1)$, where α is the exponent of the power-law fit for the starting page traffic distribution T_0: $P(T_0) \sim T_0^{-\alpha}$. To match the empirical exponent $\alpha \approx 1.75$, we thus set the parameter $\beta = 1.33$. The back button probability was set to $p_b = 0.5$ based on an analysis of the pattern of attachments to the logical sessions for the empirical data.

The ABC model contains some additional parameters not present in the BookRank model: the initial energy E_0, the forward and backward costs c_f and c_b, and the topical locality parameter η. The initial energy and the cost parameters are closely related, and they combine to control session durations. We therefore set $E_0 = 0.5$ arbitrarily and use an argument based on energy balance to find suitable values of the costs. Empirically, we know that the average size of a session is close to two pages. The net loss per click of an agent is given by $-\delta E = p_b c_b + (1 - p_b)(c_f - \langle\Delta\rangle)$, where $\langle\Delta\rangle = 1$ is the expected value of additional energy from visiting a new page. By setting $c_f = 1.0$ and $c_b = 0.5$, we obtain an expected session size of $1 - (1 - p_b)E_0/\delta E = 2$ (counting the initial page). In general, higher costs will lead to shorter sessions and lower entropy.

Setting an appropriate value for the topical locality η is a more difficult task, necessitating a number of initial simulations to explore the sensitivity of the model to the parameter η, finally settling on $\eta = 0.15$. Smaller values of η mean that all pages have similar relevance, and the session size and depth distributions become too narrow. Large values imply more noise (the absence of topical locality), and the session distributions become too broad.

The results presented here all refer to this combination of parameters ($\beta = 1.33$, $p_b = 0.5$, $E_0 = 0.5$, $c_f = 1.0$, $c_b = 0.5$, and $\eta = 0.15$). The number of users in the simulation and the number of sessions for each user are again taken from the empirical data. Because the model is computationally intensive, we partitioned the simulated users into work queues of roughly equal session

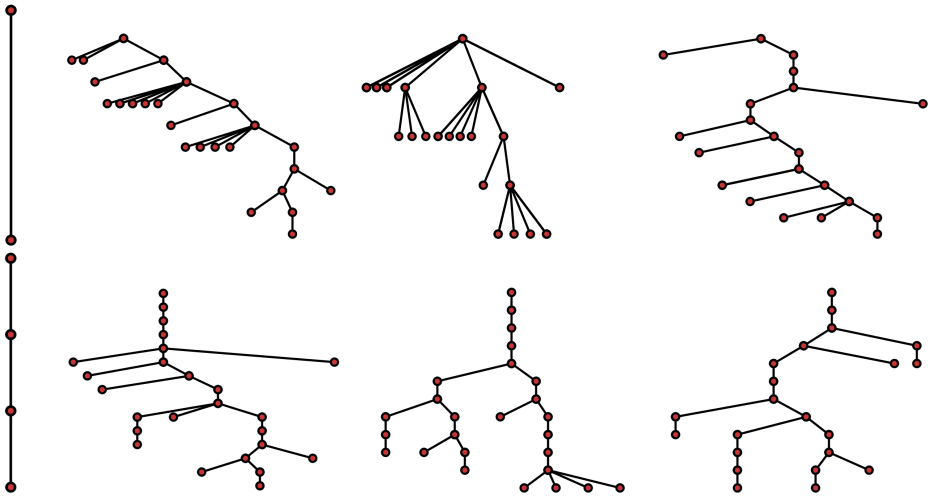


Figure 4.45: Representation of a few typical and representative session trees from the empirical data (top) and from the ABC model (bottom). Browser-based animations are available online at `cnets.indiana.edu/groups/nan/webtraffic`, using software described in Appendix A.

counts, which we executed in parallel on the Quarry system, a high-performance computing cluster operated by Indiana University Information Technology Services.

The simulations of the ABC model do yield session trees that can be compared visually to those in the empirical data, as shown in Figure 4.45. For a more quantitative evaluation of the model, we compare its results with the same empirical findings used in the evaluation of the baseline models. For each of the distributions, we also compare ABC with the reference BookRank model, which was simulated on the artificial G1 network.

The first aspect to check is whether the ABC model is able to reproduce the global features of the traffic distributions (T, ω, and T_0). Figure 4.4 shows the distribution T of the number of visits received by each page, and we can see that agreement between the ABC model and the empirical data is as good as or better than for the BookRank reference model. Similarly, the distributions of link traffic ω (shown in Figure 4.4) and starting page traffic T_0 (shown in Figure 4.4) show that the ABC model continues to reproduce the empirical data as accurately as BookRank does.

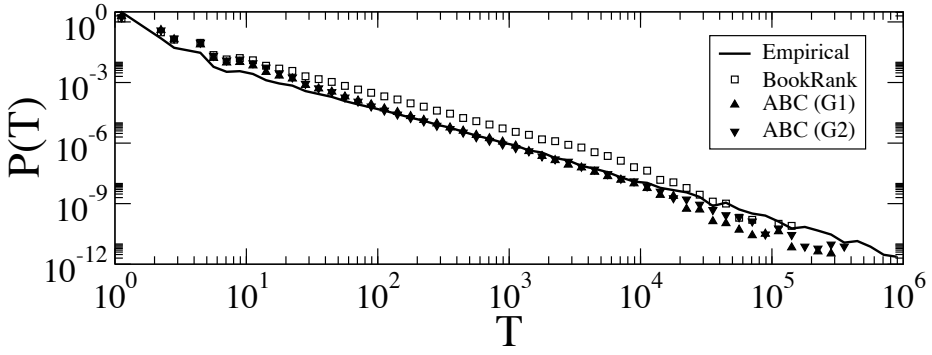


Figure 4.46: Distribution of page traffic T for ABC, empirical data, and BookRank.

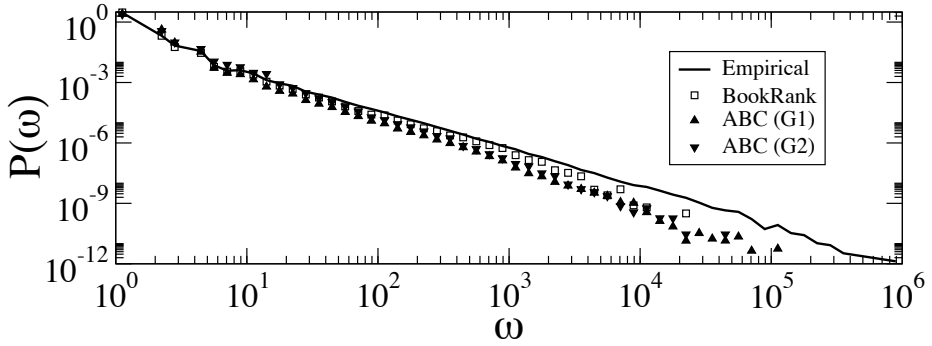


Figure 4.47: Distribution of link traffic ω for ABC, empirical data, and BookRank.

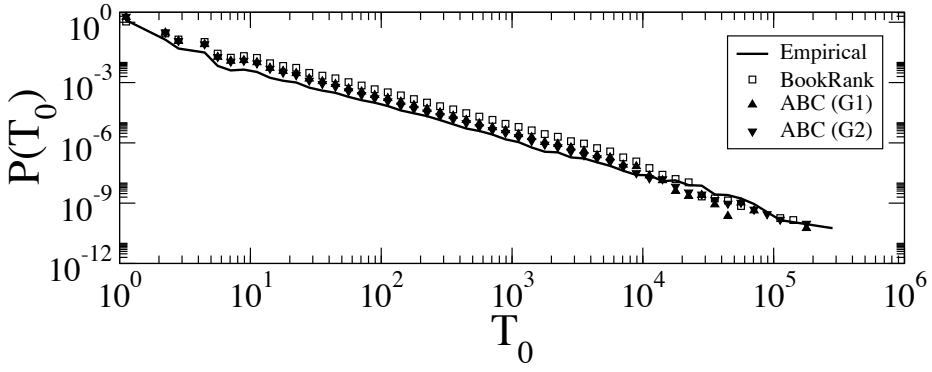


Figure 4.48: Distribution of starting page traffic T_0 for ABC, empirical data, and BookRank.

The good agreement between both the BookRank and ABC models and the empirical data provides further support for the hypothesis that the rank-based bookmark selection is a sound cognitive mechanism for capturing session initiation in Web browsing.

We can now turn our focus to how well the ABC model captures the behavior of single users. The entropy distribution across users is shown in Figure 4.4, where the predictions of ABC and BookRank are compared with the distribution derived from the empirical data. The ABC model yields entropy distributions that are somewhat sensitive to the underlying network, but that in any case fit the empirical entropy data much better than either PageRank or BookRank, in terms of both the location of the peak and the variability across users. This result suggests that bookmark memory, the back button, and topicality are all vital ingredients in explaining the focused behavior of real users.

It is also clear that the location of the peak in the entropy distribution is a more difficult property to reproduce than the shape of the traffic distribution. The entropy distribution is sensitive to a number of factors, such as the idiosyncrasies of individual users and the design of their favorite sites. We also note that the ABC model, like the PageRank random walkers, does not distinguish between outgoing links from a page (as the PRW model explored). Nevertheless, the ABC results do not fall far from matching the empirical distribution.

Having characterized traffic patterns from aggregating across user sessions, the time has come to study the sessions individually and analysis their statistical properties. Figure 4.4 depicts the distribution of the size of session trees as generated by the ABC model. The ingredients of simulated user interest and topical locality account for the broad distribution of session size, capturing the distribution of the empirical data much better than the exponentially short sessions generated by the BookRank reference model. Agents visiting relevant pages tend to keep browsing, and relevant pages tend to lead to other interesting pages, explaining the occasional very long sessions.

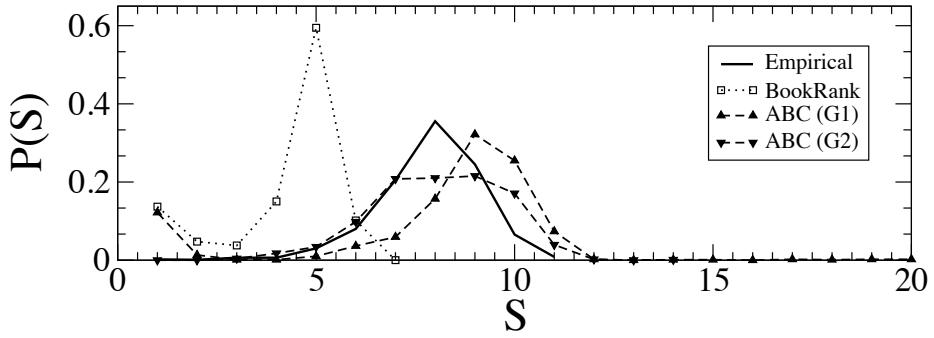


Figure 4.49: Distribution of Shannon entropy S for ABC, empirical data, and BookRank.

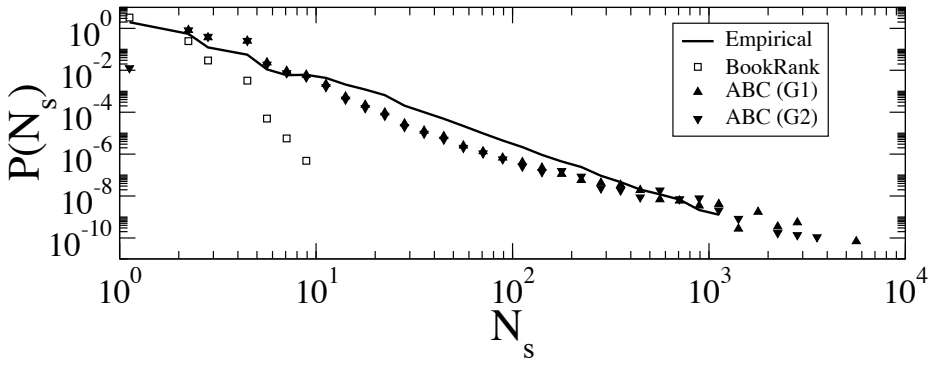


Figure 4.50: Distribution of session size N_s for ABC, empirical data, and BookRank.

This suggest that the diversity apparent in the aggregate measures of traffic is a consequence of this diversity of individual interests rather than the behavior of extremely eclectic users who visit a wide variety of Web sites—as shown by the narrow distribution of entropy.

The entropy distribution discussed above depends not only on the length of session, but also on how far each agent navigates away from the initial bookmark from which the session was initiated. One way of analyzing this is to study the distribution of session depth N_d, as shown in Figure 4.4. The agreement between the empirical data and the ABC model is again significantly better than the one observed with the BookRank baseline. We can see once more that topicality is a key ingredient in understanding real user behavior on the Web.

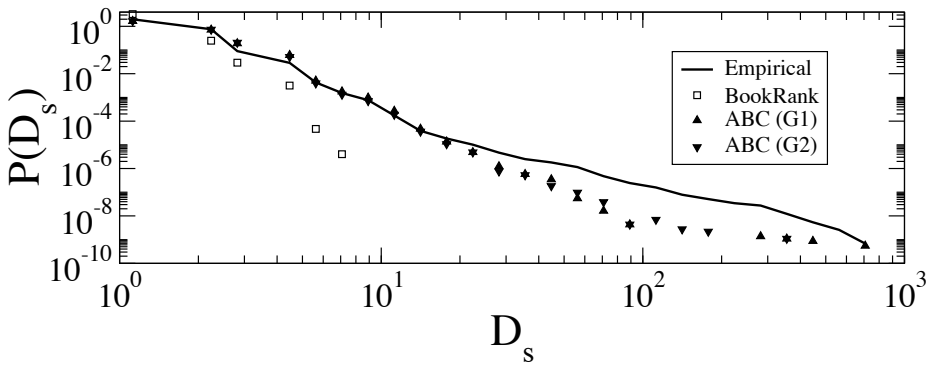


Figure 4.51: Distribution of session depth N_s for ABC, empirical data, and BookRank.

We can perform one final check on the ability of ABC to replicate real data by examining the property of assortativity once more. We have already seen in the case of both the Web behavioral network and the host graph that the structure of the network is disassortative, but that the pattern of user activity on that substrate is significantly more assortative. We can therefore hope for ABC to produce the same effect. Figure 4.4 shows a comparison of assortativity in both degree and strength for the empirical data and the G1 (artificial) and G2 (November 2009) networks. The results are mixed: we see no noticeable reduction in disassortivity between structure and behavior for the G2 network, but we can observe a small effect in the G1 network. This artificial network is perhaps the best benchmark for observation, since its random generative process creates a link structure that is neither assortative nor disassortative.

The considerable accuracy of the ABC model in reproducing the aggregate traffic patterns from the empirical data, together with the incremental means of developing and testing the model, suggest a vital role for the three novel ingredients of the model:

1. a memory mechanism, in the form of the bookmark lists;

2. a branching mechanism, in the form of the backtracking behavior; and

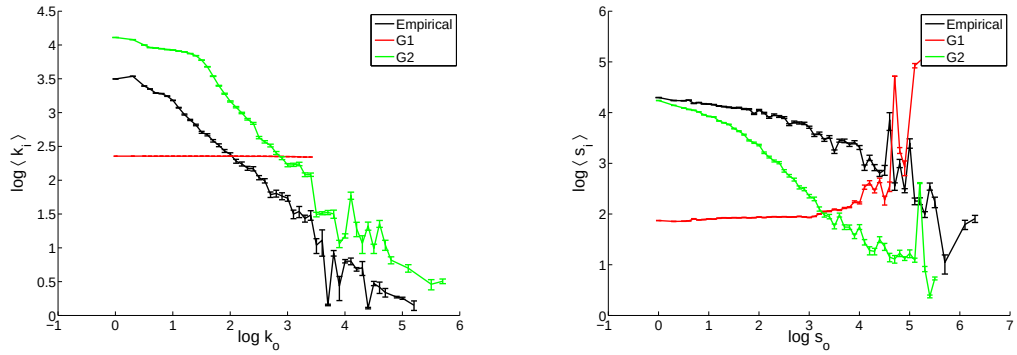


Figure 4.52: Effect of the ABC model on the assortavity of traffic, as depicted by comparing assortativity of degree or structure (left) and strength or traffic (right) for both the empirical network and the G1 and G2 networks. The G1 and G2 traffic data are generated by ABC and suggest slight movement toward greater assortativity.

3. a model of topicality, in the form of the energy-based agents.

Moreover, this model offers the first account of a mechanism that can generate key properties of logical sessions. This provides material support for the assertion that the diversity found in Web traffic is a consequence of the diversity of interests rather than users with broad and unfocused behavior. We find that Web surfers are not a collection of wildly erratic people, but a wildly erratic collection of idiosyncratic people.

The ABC model is, of course, significantly more complex than prior models such as PageRank, and its simulation is computationally intensive. However, its strong predictive power suggests that bookmarks, tabbed browsing, and topicality are salient features that deserve a role in any serious attempt to model Web surfing in a behaviorally plausible way. Besides the descriptive and explanatory power of an agent-based model with these features, these results suggest a way forward for more sophisticated, realistic, and effective ranking and crawling algorithms.

The ABC model does rely on a number of key parameters. While this evaluation is based on

choices for these parameters that seem reasonable and realistic, and I have performed some empirical exploration of parameter space to better understand the sensitivity of the model, further work is needed to achieve a complete understanding of their effect on overall behavior. For example, it is already clear that some parameters such as the network size, movement costs, and topical locality factor play a key role in modulating the balance between individual diversity (entropy) and session size.

I believe that the analysis of the traffic-weighted Web graph, development of the notion of logical sessions, and development of an improved model of user behavior on the Web represent a substantial step forward in our ability to understand the structure and dynamics of the dominant application on the Internet.

5

Conclusion

My journey through the world of Internet traffic has been a long and winding one. At the outset, my hope had simply been that taking a graph-based approach to viewing network data would provide a computationally efficient way of identifying anomalies not readily apparent under the relational view of flow data. It soon became clear from the results in Chapter 3 that Internet traffic does not provide a background level of normal behavior from which abormal activity can be distinguished. Extreme heterogeneity can be seen in nearly every aspect of network activity, implying that because human behavior online is largely unconfined from the restraints of the physical world, we lose any concept of normalcy. No matter how busy a Web server is, there is probably one busier. No matter how much data a single host has traded on Bittorrent, there is probably one who has traded more.

This lack of regularity against which any anomalies stood out forced me to consider what graph analysis could provide beyond negative results that imply the inefficacy of a large swathe of security products, which led directly to the work of application classification. The success in identifying unknown applications proved that it was possible to obtain useful results from a graph-theoretic

approach that respected user privacy and avoided the computational pitfalls of deep packet inspection.

My attention turned to the analysis of Web clicks in response to my frustration at the inability of network flow records to include any application protocol information. I quickly realized that referrer information made an HTTP request far more than a record of a transaction between a client and server, but rather a record of traffic passing from one server to another. This led to the construction of the weighted host graph and the investigation of the ways in which PageRank fails to predict its properties.

The opportunity to collect click information that maintained client identities offered an opportunity to better understand where all of these long-tailed and scale-free distributions come from: are individual network users really so pathologically eclectic? The discovery of normal and log-normal distributions at the level of individual users implied that the extreme distributions we see in Web traffic are more a consequence of diverse types of users than of extreme variance in the behavior of single users.

This intuition, coupled with the development of logical sessions as a basis for understanding individual patterns of interaction with the Web, paved the way for the development of the BookRank and ABC models. The ABC model stands as the culmination of this effort: a behaviorally plausible but simple model that shows how the individual actions of a diverse community of users can combine to yield aggregate distributions so wide as to lack any central tendency.

In the remainder of this chapter, I summarize the major findings and contributions of the work, first in the context of network flow analysis and then in the context of Web click analysis. I then briefly discuss some open questions and possible future directions for this research.

Throughout this work, I have endeavored to remain true to the themes expressed in Chapter 1.

This research has consistently been concerned with how things actually behave rather than how they are defined or their ostensible purpose. Applications are classified by how they act, not how they are written. The physical link structure of the Web is less interesting than the version weighted by traffic because the links that people actually use are more truly a part of the Web than the ones they ignore. My use of network analysis rather than traditional relational data mining techniques is based on my abiding belief in the importance of patterns of interaction over the study of entities in isolation. This emphasis has been at times indispensible; for example, without doing the global reshuffling of weights in the host graph, it would have been impossible to attribute most of the diversity in link traffic within a site to the overall distribution rather than genuine local effects. All of the work has been computationally tractible and the data gathered using commodity hardware. Although the click data did in the end involve packet inspection, it did so in the most efficient and scalable way possible, achieving high rates of collection on a single low-cost server. Finally, all of this work has been performed with a manner mindful of human dignity and user privacy—both the flow records and clicks were anonymized, and there has been neither any storage nor breach of sensitive personal information. The fact that so many compelling results can be coaxed from the data notwithstanding this emphasis is a reminder that we are often too quick to assume that we must sacrifice our privacy if we are to understand the Internet and keep it safe.

5.1 Summary of results

This section summarizes the key findings and novel contributions from the studies and experiments described in Chapters 3 and 4.

Network flow analysis

The preliminary study of network flow data that examined the behavioral network of the Web in isolation was the first evidence of scale-free distributions of degree and strength in Internet traffic data. The study found that the distributions of client degree and strength were well-approximated by power laws of the form $P(x) \sim x^{-\gamma}$ with values of $gamma < 3$, implying unbounded variance. This study also found a power-law distribution with $gamma < 3$ for the volume of interactions between clients and servers and demonstrated a superlinear scaling relationship between degree and strength for Web clients. Finally, it showed that degree and strength distributions for Web servers obeyed power-law approximations with $gamma < 2$, indicating that both the variance and mean were unbounded and showing the lack of any meaningful central tendency in the data.

The extended study of network flow traffic that compared properties of several classes of Internet data (Web, P2P, and other) across three years of network evolution was unprecedented in its detail. The results indicated a significantly smaller overlap between the sets of client and server hosts for Web traffic than for P2P and "other" traffic, providing evidence that much traffic unclassifiable by port number is still related to P2P networks. The study affirmed the long-term stability of the findings of the previous study in regard to Web traffic and uncovered a definite exponential cutoff for P2P traffic, likely a result of limited host capacity and human attention. The superlinear relationship between strength and degree for Web clients was confirmed.

The study of application networks and subsequent hierarchical clustering of network applications based on observed behavior provided evidence that anonymized network flow records provide sufficient evidence for identifying and classifying unknown applications. The clustering technique was successful in guessing the identities of a majority of unknown applications in the two trials, and no case was it proven to be incorrect. It revealed features of the evolution of Bittorrent traffic and was able to recover the identity of at least one application whose packet payloads

would have been undecipherable to an analyst unfamiliar with the Korean language.

The study of bias introduced through the sampling of packet data in constructing network flow records found that although the mechanism of sampling does seem to match the descriptions of router vendors, it also introduces a systematic bias toward smaller flows. Regarding flows as tuples causes us to believe that large flows occur less frequently than they actually do and leads to an underestimate of the variance of the distribution. The aggregation technique used to build up the graphs involved in Chapter 3 is much less vulnerable to this bias, serving as another advantage of a graph-centric view of flow data.

Web click analysis

The initial study of HTTP requests described in Chapter 4 was the largest of its kind in the literature, and data collection is still ongoing as of this writing, making it among of the richest sources of Web request data available to the research community. Additionally, the data are anonymized and gathered in a way that minimizes the sources of bias attendant to many other means of collecting Web traffic data.

This study revealed many heavy-tailed distributions among the properties of the traffic-weighted Web host graph. Both the in-degree and out-degree of sites were found to be well-approximated by power laws of the form $P(x) \sim x^{-\gamma}$ with $gamma < 3$, implying unbounded variance. The study also found a systematic difference in estimates of in-degree derived from traffic data and Web crawls, making it a philosophical issue as to which should be considered ground truth. The distributions of strength and link traffic were fit with power-law models with $\gamma < 2$, implying the lack of any central tendency and suggesting that Web traffic has properties inaccessible to any analysis of the static link graph.

The study also found a that a surprisingly small proportion of Web traffic can be attributed directly to search engines. However, it also found that traffic eminating from search engines was more likely to lead to novel sites, implying that rather than restrict traffic to a shrinking volume of popular sites ("Googlearchy"), search engines help users discover sites they would not find through surfing alone.

With respect to temporal properties of Web traffic, the study found that although Web traffic observes strong daily, weekly, and semester-long cycles, the relative proportion of Web traffic attributable to various categories of site is largely stable. It also found that a large proportion of Web traffic (over 30%) is essentially static, and that the baseline prediction of Web traffic—that the future will be exactly like the present—is quite difficult to best.

Finally, the study examined the three assumptions of uniform distributions underlying the PageRank random surfer model (homogeneity of links, starting points, and jumping point) and found each of them to be contradicted by actual traffic data. The diversity of traffic among the outgoing links of a page was found to account for only a minor part of PageRank's inability to predict traffic, as shown by the PRW model. This was explained through use of the Herfindahl-Hirschman index and randomly shuffled weights. The study found that starting page traffic observes extreme heterogeneity and is well-approximated by a power-law, directly contradicting the uniform assumption of PageRank. Finally, it found that not all sites are equally likely to be teleporation points: some sites produce more traffic than they receive (hubs), and some absorb nearly all incoming traffic (sinks).

The dormitory study, which maintained a distinction among the requests of different users, was the foundation of the remaining results. Analysis of its weighted host graph replicated the findings of previous studies, confirming scale-free distributions of traffic. It also showed that the popularity of sites as measured by the number of distinct users obeys a wide power-law distribution. Most

per-user distributions, such as the number of requests, number of empty-referrer requests, and request rate, were found to be log-normal, showing the first evidence of actual central tendencies in distributions related to Web traffic. The analysis also identified two classes of Web user: one more characterized by searching, and one more characterized by surfing.

A primary contribution of the host-level analysis of the dormitory data was the finding that segmentation of individual users' click streams into sessions based on an inactivity timeout is arbitrary and ineffective in constructing a meaningful definition of a Web session. The distribution of times between clicks was found to be scale-free for every user, implying that regularity is the exception rather than the rule. The logical sessions proposed are more robust, correspond intuitively to the behavior of users on modern browsers, and display per-user central tendencies that are both well-defined and plausible. They also provide strong evidence that the attention of users is split as they navigate the Web, implying the widespread use of the back button and multiple tabs. Finally, logical sessions were shown to be fairly insensitive to the introduction of a timeout of sufficient duration.

The BookRank model was an attempt to fix the deficiencies of the PageRank random surfer model by introducing state in the form of a bookmark list that reflected previous traffic. It was shown to estimate site and link traffic better than PageRank, but it did not include a mechanism capable of reproducing the wide variation in session length and depth found in empirical data.

The ABC model extended the BookRank model with stateful agents that incorporated an energy-based model of topical locality. This model was found to meet or exceed the performance of both PageRank and BookRank in predicting six properties of Web traffic: page traffic, link traffic, empty referrer traffic, entropy, session size and session depth. The sessions produced by the ABC model are comparable to empirical sessions and their means of generation is behaviorally plausible. I believe ABC to be the most powerful and complete model of Web traffic available to date.

5.2 Future work

I have attempted to identify some possible avenues of future research throughout this work, which I now summarize.

In the domain of network flow analysis, although spectral analysis of the data seems to be at an impasse, researchers more experienced in the field may be able to extend this work in ways I cannot anticipate. A particularly fertile area of future research for behavioral networks may be the identification of particular motifs—for example, using unusually symmetric roles for hosts as evidence that the hosts may be operating as open proxies. If unsampled flow data becomes more widely available, behavioral networks may be useful in identifying the use of various advanced anonymization techniques such as "port-knocking", in which a sequence of attempted connections are prerequisite to establishing a real conversation. My analysis can also extended to other network properties as computing power increases and efficient heuristic algorithms improve; measurements such as betweenness centrality may be more useful in uncovering anomalous hosts. It may also be possible to glean interesting results from the bipartite behavioral networks by studying their unipartite co-citation networks: for example, the network formed by connecting Web servers that share common clients.

The analysis and modeling of Web traffic is an important and quickly expanding field, but I can identify some specific suitable starting points for extending the research described here. One potential avenue of research is to attempt to improve on the baseline prediction for forecasting Web traffic by combining topological information with the history of weights across links. Future investigation is strongly warranted when it comes to the role of search engines in mediating Web traffic; the division of users into "search" and "surf" modes and the unexpectedly low proportion of traffic generated directly by search engines suggests there are many stories yet to be told. The

traffic data also provide an opportunity for validating the HITS algorithm by seeing whether it is able to predict the sites that generate more traffic than they receive.

Although the ABC model seems to work well, it is not a finished work. Not all of its parameters are well-understood, and the parameter space needs to be explored in much greater detail. It would profit greatly from comparison to another set of empirical traffic data with similar qualities, especially if more users were involved. Finally, as an agent-based model, it is necessarily computationally expensive to simulate for a large user population. Any improvements to the model or its actual software implementation that make it more efficient while maintaining its predictive power and behavioral plausibility would be immensely beneficial.

Finally, it must be said that this journey did not end in the place I expected when I first set out, but it had yielded many insights and observations that I hope can improve our understanding of the Internet and the people who use it. The most promising future work—and, in the end, the greatest contribution of this research—lies in the directions I cannot yet imagine but that are inspired by some aspect of what I have presented here.

A

Software

A.1 Flow collection

All network flow collection is performed by the *indexer* program, which is a C application capable of anonymizing a high-volume stream of *netflow-v5* data and streaming in to disk in real time. It is built on top of the standard *libpcap* packet capture library[100] and is capable of gathering data from any active network interface or port. Before each flow is written to disk, *indexer* performs a number of actions for protecting user privacy and easing later analysis:

1. The source and destination IP addresses are replaced with first-come, first-served numeric index values, in accordance with Internet2 privacy policy. The mapping between IP addresses and index values is stored only in system memory and is never recorded to disk.

2. A copy of the source and destination IP addresses are anonymized by zeroing out the lower 11 bits, in accordance with Internet2 privacy policy.

3. The *first* and *last* fields are replaced by the relevant Unix epoch in seconds, calculated using the *unix_secs* and *unix_nsecs* fields from the packet header.

4. The fields are rearranged as indicated in Figure A.1, reducing the size of each flow record from 48 bytes to 42 bytes.

Besides these actions, *indexer* also maintains a record of the highest index assigned and flow tallies for each TCP and UDP port, which are used for the heuristic assignment of flow directionality.

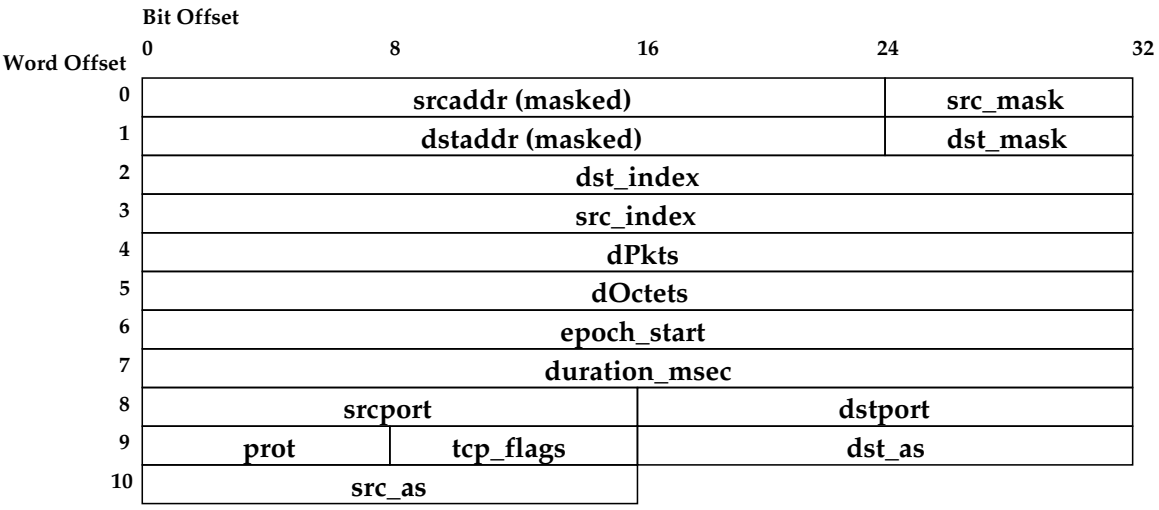


Figure A.1: Format of an anonymized network flow record as written by *indexer*.

The operation of *indexer* is managed by the Perl script *capture-data*, which uses external configuration files to select various possible sources of flow data. This script handles segmenting the flow data into multiple files on periodic boundaries and aborts collection if the filesystem runs too low on disk space.

A.2 Flow exploration

The initial exploration of network flow data was made possible through a suite of personally developed tools called *flowseek* that make it possible to search a large volume of flow records efficiently using a simple but effective query language that supports Boolean and relational operators. The impetus for development was the cryptic interface and poor performance of existing flow analysis software such as *flow-tools* [67], which incorporates a pipeline-based design that causes flow data to be translated between binary and ASCII format several times during many operations. The overarching goal of *flowseek* is to be fast enough to support interactive, exploratory operation even

on large volumes of data.

The tool suite is written entirely in standard C and should be easily portable to any POSIX system with support for dynamic object files. Although all development was done on an Intel-based system, the software makes no assumptions about native byte order. In order to make the system as modular as possible, the majority of *flowseek* is found in three libraries that provide a standard API for command-line tools:

- *libformat*

  A major drawback of *flow-tools* is the difficulty in selecting an appropriate output format; one is either forced to pick a format from a small list of choices or master a complex set of configuration files. The usual solution is to write many small throw-away scripts to massage data into an appropriate format. The *libformat* library avoids this by providing an easy way for command-line applications to give the user full control over output formatting.

- *libdumpfile*

  Some flow analysis tools can only examine flow data directed to an active socket on the analysis system, requiring the simulated retransmission of the data in order to process it. To avoid this, the *libdumpfile* library provides routines for extracting structured data from *libpcap* capture files and providing it to client applications. A system of dynamic plug-ins makes the library applicable for many purposes beyond flow analysis.

- *libflowseek*

  The library is the computational core of the *flowseek* system. It contains routines that accept a user query, reduce it to a boolean and arithmetic expression, produce C code implementing the query, compile that code, and dynamically import it into a running application. (The speed of the system stems from this dynamic importation.)

The query language implemented by *flowseek* is strongly reminiscent of the Berkeley Packet Filter language and various advanced search engines. In its basic form, a query contains arithmetic comparisons and built-in functions that are aggregated using parentheses and the standard boolean operators. For example, a query that seeks to find all flow records in which a host in the subnet 129.79.0.0/16 was directing traffic to TCP port 80 of a remote host might be rendered as

```
src IN("129.79.0.0", 16) AND (proto = 6) AND (dst_port = 80)
```

Each query is subject to a series of simplification steps that reduce it to a sum-of-products expression in which the factors of each term are simple arithmetic tests. The orders of the factors and terms are then reordered heuristically to speed up short-circuiting evaluation. The query is rendered into C code, compiled as a shared library, and dynamically loaded into the running application using standard system libraries.

A.3 Flow generation

As described in Chapter 3, the Flow Generation Language (FGL) provides an easy way to automate the introduction of complex patterns of traffic to the network. Programs are executed in two parts. First, an FGL script is processed by the interpreter, which is written in Perl and generates as output a network event file. The event player application is written in C and uses the network event file to generate actual packets, which are then injected directly into the network using a raw socket interface.

The syntax of the FGL language is well-suited for the task of generating complex traffic patterns. It supports a wide variety of built-in functions, procedure declarations, and named parameters. More importantly, it supports lists, sets, and procedures as first-class data types. Procedures can be easily be passed as parameters or return values and generated on the fly. They are stored using

```
println("Bias study #3 (2008-12-10)");
println();
println("This FGL code will generate 10 256-byte packets to each UDP port");
println("in the range 10000-10009 on the hosts 64.57.17.200 - 64.57.17.209.");
println();

x = proc(pkt) begin
  println("Emitting 10 of ", pkt);
  notate(pkt);
  emit(pkt, 10, 0.1);
  delay(0.10);
end;

port    = range(10000, 10009);
host    = range(start:ip("64.57.17.200"), end:ip("64.57.17.209"));

xip     = [ ip_header(src:ip("156.56.103.1"), dst:@host) ];
xudp    = [ udp_header(src_port:0, dst_port:@port) ];
xpacket = [ udp(@xip, @xudp, size:256, data:"This is a test.") ];

output("bias-study-3.event");
x(@xpacket);
```

Figure A.2: Example FGL script.

closures, which allows the interpreter to avoid confusion through strict call-by-value semantics.

The most important feature of the language is the expansion operator @, which enables implicit iteration. Whenever a list or set expression is prefixed with the expansion operator, the interpreter will evaluate the surrounding expression once for every element in the list or set. If multiple expansion operators are found in a single expression, the expression is evaluated once for every tuple in the Cartesian product of the elements of each expanded expression. This makes it possible to generate an extremely large number of packets which vary in five different ways using a single line of FGL code and no explicit loops.

A sample FGL script is included in Figure A.2 to offer a flavor of the language and its capabilities.

A.4 Large graph analysis

As discussed in the text, working with enormous graph structures and data files, some of them too large to fit in main memory, has necessitated the development of some homegrown tools for working with these files. A complete listing of these tools, which are a collection of command-line applications with some basic library support, would be inappropriately detailed in this context. This suite of tools is mentioned here primarily to advertise its existence; I am working to tidy their interface and make them available to the general community through an open-source license.

A.5 Click collection

The gathering of HTTP requests is performed through the *clicksniffer* application, which written in C and optimized to parse well-formed HTTP requests out of a stream of network data as quickly as possible. The application uses the *libpcap* library [100] to install a Berkeley Packet Filter matching TCP destination port 80; this filter will be executed in kernel space on a FreeBSD system. Matching packets are subjected to a series of precompiled regular expressions to extract the relevant features from packets that contain requests.

A number of configuration options are available to adapt the application to different collection purposes. For example, the sniffer can be made to retain the IP and MAC addresses of the system generating the requests, as was done for the dormitory study. Other options include the ability to tune the capture loop of *libpcap*, include directionality information particular to IU, include a flag for user agents corresponding to common browsers, reject requests with no referrer, and log only the host portion of each referring URL.

As in the case of the *indexer* flow collection system, an associated Perl script handles the task of

segmenting the incoming data into separate files at periodic intervals.

A.6 Traffic modeling

The code for the implementation of the ABC model was developed jointly by myself and Bruno Gonçalves, and we anticipate its eventual release to the wider research community. The simulation engine allows the specification of all model parameters on the command line and is instrumented to produce a wealth of runtime data for offline analysis. Each run of the model application simulates a quantity of sessions for a single user and generates three output files:

- A *session* file that contains the values of the parameters used and a record of all moves and jumps taken by the agent. At the conclusion of each session, its duration in time steps, size, and depth are also recorded in this file.
- A *traffic* file that records the final contents of the ranked bookmark list and the number of times each of those nodes has been visited. Because the bookmark list is always updated for the first visit to a node within a session, this is the basic measurement of generated traffic.
- An *energy* file that records the remaining energy of the agent at each time step, making it possible to study the rise and fall of energy for very long sessions.

The ABC model is coupled with a suite of scripts written in Perl and MATLAB that automate the process of running the model for a collection of users and graphing the distributions mentioned in the text. These tools can also ease the process of dividing a simulation across multiple nodes in a high-performance computing cluster.

A.7 Session visualization

I have also produced a Processing application called *Nursery* for interactively visualizing the growth of both empirical sessions and those generated by the ABC model. This application is available online from the Web traffic analysis group home page [1], which also provides access to the source code for the visualization.

The application includes a novel algorithm for making sure that trees are visually balanced on the screen and nodes stay in a fixed location throughout the animation.

[1] http://cnets.indiana.edu/groups/nan/webtraffic

Bibliography

[1] S. Acharya, B. Smith, and P. Parnes. Characterizing user access to videos on the world wide web. In *SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, 1998.

[2] D. Achlioptas, A. Clauset, D. Kempe, and C. Moore. On the bias of traceroute sampling. In *STOC*, 2005.

[3] L Adamic and BA Huberman. Power-law distribution of the World Wide Web. *Science*, 287:2115, 2000.

[4] Lada A. Adamic and Bernardo A. Huberman. The Web's hidden order. *Communications of the ACM*, 44(9):55–60, 2001.

[5] Eytan Adar, Jaime Teevan, and Susan Dumais. Large scale analysis of web revisitation patterns. In *Proc. CHI*, 2008.

[6] Eytan Adar, Jaime Teevan, and Susan Dumais. Resonance on the Web: Web dynamics and revisitation patterns. In *Proc. CHI*, 2009.

[7] E. Agichtein, E. Brill, and S. Dumais. Improving Web search ranking by incorporating user behavior information. In *Proc. 29th ACM SIGIR Conf.*, 2006.

[8] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing*, pages 171–180. ACM, 2000.

[9] R Albert, H Jeong, and A-L Barabási. Diameter of the World Wide Web. *Nature*, 401(6749):130–131, 1999.

[10] Alexa. Alexa, 2005. http://pages.alexa.com/prod_serv/data_services.html.

[11] E. Almaas, B. Kovacs, T. Vicsek, Z. N. Oltvai, and A.-L. Barabasi. Global organization of metabolic fluxes in the bacterium Escherichia coli. *Nature*, 427(6977):839–843, 2004.

[12] Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. *Large scale networks fingerprinting and visualization using the k-core decomposition*, pages 41–50. MIT Press, Cambridge, MA, 2006.

[13] M. Arlitt and C. Williamson. Web server workload characterization: the search for invariants. In *ACM SIGMETRICS*, 1996.

[14] R Baeza-Yates, F Saint-Jean, and C Castillo. Web structure, dynamics and page quality. In Alberto H. F. Laender and Arlindo L. Oliveira, editors, *Proc. 9th Intl. Symp. on String Processing and Information Retrieval (SPIRE 2002)*, volume 2476 of *Lecture Notes in Computer Science*, pages 117–130. Springer, 2002.

[15] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[16] A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: The toplogy of the World Wide Web. *Physica A*, 281:69–77, 2000.

[17] A. Barrat and M. Weigt. On the properties of small-world networks. *Eur. Phys. J. B*, 13:547–560, 2000.

[18] Marc Barthélemy, Bernard Gondranb, and Eric Guichardc. Spatial structure of the Internet traffic. *Physica A*, 319:633–642, March 2003.

[19] Thomas Beauvisage. The dynamics of personal territories on the web. In *HT '09: Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 25–34, New York, NY, USA, 2009. ACM.

[20] M. G. Beiró, J. I. Alvarez-Hamelin, and J. R. Busch. A low complexity visualization tool that helps to perform complex systems analysis. *New J. Phys.*, 10:125003, 2008.

[21] Mary Bellis. Intel 4004: The world's first single chip microprocessor. http://inventors.about.com/od/mstartinventions/a/microprocessor.htm, 2009.

[22] Laurent Bernaille, Renata Teixeira, and Kavé Salamatian. Early application identification. In *CoNEXT*, 2006.

[23] K. Bharat, B.-W. Chang, M. Kenzinger, and M. Ruhl. Who links to whom: Mining linkage between web sites. In *Proceedings of First IEEE International Conference on Data Mining (ICDM'01)*, 2001.

[24] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. Do your worst to make the best: Paradoxical effects in pagerank incremental computations. *Internet Mathematics*, 2(3):387–404, 2005.

[25] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. Pagerank as a function of the damping factor. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 557–566, New York, NY, USA, 2005. ACM Press.

[26] Kevin Borders and Atul Prakash. Web tap: Detecting covert Web traffic. In *In Proceedings of*

the 11th ACM Conference on Computer and Communication Security, pages 110–120. ACM Press, 2004.

[27] M. Bouklit and F. Mathieu. BackRank: an alternative for PageRank? In *Proc. WWW Special interest tracks and posters*, pages 1122–1123, 2005.

[28] Lee Breslau, Pei Cue, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *In INFOCOM*, pages 126–134, 1999.

[29] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1–7):107–117, 1998.

[30] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the Web. *Computer Networks*, 33(1–6):309–320, 2000.

[31] A. Broido and K. C. Claffy. Internet topology: Connectivity of IP graphs. In *San Diego Proceedings of SPIE International Symposium on Convergence of IT and Communication*, 2001.

[32] The Cooperative Association for Internet Data Analysis. http://www.caida.org/home/.

[33] CAIDA.org. cflowd: Traffic flow analysis tool. http://www.caida.org/tools/measurement/cflowd/.

[34] CAIDA.org. Flowscan: Network traffic flow visualization and reporting tool. http://www.caida.org/tools/utilities/flowscan/.

[35] L. Carbone, F. Coccetti, P. Dini, R. Percacci, and A. Vespignani. The spectrum of Internet performance. In *Proceedings of Passive and Active Measurement (PAM2003)*, 2003.

[36] Lara D. Catledge and James E. Pitkow. Characterizing browsing strategies in the World-Wide Web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, 1995.

[37] H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. Towards capturing representative AS-level Internet topologies. *Computer Networks Journal*, 44:737–755, April 2004.

[38] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. The origin of power laws in Internet topologies revisited. In *Proceedings of IEEE Infocom*, 2002.

[39] L. Cherkasova and M. Gupta. Analysis of enterprise media server workloads: Access patterns, locality, content evolution, and rates of change. In *IEEE/ACM Journal on Transactions of Networking (ToN)*, pages vol. 12, pages 781–794, 2004.

[40] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through URL ordering. *Computer Networks*, 30(1–7):161–172, 1998.

[41] Junghoo Cho and Sourashis Roy. Impact of search engines on page popularity. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proc. 13th intl. conf. on World Wide Web*, pages 20–29. ACM, 2004.

[42] K. C. Claffy. Internet measurement and data analysis: topology, workload, performance and routing statistics. In *NAE '99 workshop*, 1999. CAIDA.

[43] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51:661–703, 2009.

[44] Andy Cockburn and Bruce McKenzie. What do Web users do? An empirical analysis of Web use. *Intl. Journal of Human-Computer Studies*, 54(6):903–922, 2001.

[45] Brian F. Cooper, Eric Baldeschwieler, Rodrigo Fonseca, James J. Kistler, P. P. S. Narayan, Chuck Neerdaels, Toby Negrin, Raghu Ramakrishnan, Adam Silberstein, Utkarsh Srivastava, and Raymie Stata. Building a cloud for Yahoo! *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2009.

[46] L. Dall'Asta, I. Alvarez-Hamelin, A. Barrat, A. Vázquez, and A. Vespignani. Exploring networks with traceroute-like probes: Theory and simulations. *Theoretical Computer Science, Special Issue on Complex Networks*, 2005.

[47] Brian D. Davison. Web traffic logs: An imperfect resource for evaluation. In *Ninth Annual Conference of the Internet Society (INET)*, 1999.

[48] Brian D. Davison. Topical locality in the Web. In *Proc. 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 272–279, 2000.

[49] S. Dill, R. Kumar, K. S. McCurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-similarity in the web. *ACM Transactions on Internet Technology*, 2(3):205–223, 2002.

[50] The DIMES project. http://www.netdimes.org/new/.

[51] D. Donato, L. Laura, S. Leonardi, and S. Millozzi. Large scale properties of the webgraph. *Eur. Phys. J. B*, 38:239–243, 2004.

[52] Allen B. Downey. Evidence for long-tailed distributions in the internet. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.

[53] Allen B. Downey. Lognormal and pareto distributions in the internet. *Computer Communications*, 28:790–801, 2004.

[54] H. Ebel, L.-I. Mielsch, and S. Bornholdt. Scale-free topology of e-mail networks. *Phys. Rev. E*, 66:035103, 2002.

[55] Dan Eggen. Under fire, justice shrinks TIPS program. *Washington Post*, August 2002.

[56] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[57] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *ACM SIGCOMM Workshop on Mining Network Data*, pages 281–286, 2006.

[58] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, and Carey Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core. In *WWW*, pages 883–892, 2007.

[59] Cristian Estan, Stefan Savage, and George Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of the ACM SIGCOMM Conference*, 2003.

[60] Réseaux IP Européens. Ripe network coordination centre. http://www.ripe.net.

[61] A. Fabrikant, E. Koutsoupias, and C. H. Papadimitriou. Heuristically optimized trade-offs: A new paradigm for power laws in the Internet. In *Proceedings of the 29^{th} International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 2380 of *Lecture Notes in Computer Science*, pages 110–122. Springer, 2002.

[62] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[63] S. Forrest, S. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, 1997.

[64] S. Fortunato, M. Boguna, A. Flammini, and F. Menczer. Approximating PageRank from in-degree. In *Proc. WAW 2006*, volume 4936 of *LNCS*, pages 59–71. Springer, 2008.

[65] S. Fortunato, A. Flammini, F. Menczer, and A. Vespignani. Topical interests and the mitigation of search engine bias. *Proc. Natl. Acad. Sci. USA*, 103(34):12684–12689, 2006.

[66] Santo Fortunato and Alessandro Flammini. Random walks on directed networks: the case of pagerank. *International Journal of Bifurcation and Chaos*, 2007. Forthcoming.

[67] Mark Fullmer. flow-tools information. http://www.splintered.net/sw/flow-tools/.

[68] Jianfeng Gao, Wei Yuan, Xiao Li, Kefeng Deng, and Jian yun Nie. Smoothing clickthrough data for web search ranking. In *Proc. SIGIR*, 2009.

[69] Michele Garetto, Weibo Gong, and Don Towsley. Modeling malware spreading dynamics. In *Proc. INFOCOM 2003, 22nd Annual Joint Conference of the IEEE Computeer and Communications Societies*, 2003.

[70] A. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In *Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 51–62, 1999.

[71] B. Gonçalves and J. J. Ramasco. Human dynamics revealed through web analytics. *Phys. Rev. E*, 78:026123, 2008.

[72] Bruno Gonçalves, Mark Meiss, José J. Ramasco, Alessandro Flammini, and Filippo Menczer. Remembering what we like: Toward an agent-based model of web traffic. In *WSDM Late-Breaking Results*, 2009.

[73] Bruno Gonçalves and José Ramasco. Modeling the behavior of individual websurfers. In *Proc. 23rd IUPAP International Conference on Statistical Physics*, 2007.

[74] Miniwatts Marketing Group. Internet world stats: Usage and population statistics. http://www.internetworldstats.com/stats.htm, 2009. Fetched July 14, 2009.

[75] N. Harel, V. Vellanki, A. Chervenak, and G. Abowd. Workload of a media-enhanced classroom server. In *IEEE Workshop on Workload Characterization*, 1999.

[76] G. Helmer, J. Wong, V. Honavar, and L. Miller. Intelligent agents for intrusion detection. In *IEEE Information Technology Conference*, pages 121–124, 1998.

[77] O.C. Herfindahl. *Copper Costs and Prices: 1870-1957*. John Hopkins University Press, Baltimore, MD, 1959.

[78] A.O. Hirschman. The paternity of an index. *American Economic Review*, 54(5):761–762, 1964.

[79] B.A. Huberman and R. Lukose. Social dilemmas and Internet congestion. *Science*, 277:535, 1997.

[80] B.A. Huberman, P.L.T. Pirolli, J.E. Pitkow, and R.M. Lukose. Strong regularities in World Wide Web surfing. *Science*, 280(5360):95–97, 1998.

[81] B. Huffaker, M. Fomenkov, D. Moore, E. Nemeth, and K.C. Claffy. Measurements of the Internet topology in the asia-pacific region. In *INET '00, Yokohama, Japan, 18-21 July 2000,*. The Internet Society, 2000.

[82] Cisco Systems Inc. Netflow export datagram format. http://www.cisco.com/univercd/cc/td/doc/ product/rtrmgmt/nfc/nfc_3_0/nfc_ug/nfcform.htm.

[83] Cisco Systems Inc. Netflow v9 export format. http://www.cisco.com/en/US/docs/ios/12_3/feature/gde/nf

[84] Internet2 project. http://abilene.internet2.edu/.

[85] Lucas Introna and Helen Nissenbaum. Defining the web: The politics of search engines. *IEEE Computer*, 33(1):54–62, January 2000.

[86] C. Jin, Q. Chen, and S. Jamin. INET: Internet topology generators. Technical Report CSE-TR-433-00, EECS Dept., University of Michigan, 2000.

[87] Thorsten Joachims. Evaluating search engines using clickthrough data. In *Proceedings of the SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, pages 133–142, 2002.

[88] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. BLINC: Multilevel traffic classification in the dark. In *SIGCOMM*, pages 229–240, 2005.

[89] M. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–89, 1938.

[90] J Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[91] Jack Koziol. *Intrusion Detection with Snort*. Sams, 2003.

[92] SR Kumar, P Raghavan, S Rajagopalan, D Sivakumar, A Tomkins, and E Upfal. Stochastic models for the Web graph. In *Proc. 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 57–65, Silver Spring, MD, 2000. IEEE Computer Society Press.

[93] A. Lakhina, J. Byers, M. Crovella, and P. Xie. Sampling biases in IP topology measurements. In *INFOCOM*, 2003.

[94] Anukool Lakhina, Mark Crovella, and Christophe Diot. Characterization of network-wide anomalies in traffic flows. In *Proceedings of the ACM/SIGCOMM Internet Measurement Conference*, October 2004.

[95] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of ACM SIGCOMM 2004*, August 2004.

[96] L. Laura, S. Leonardi, S. Millozzi, U. Meyer, and J. F. Sibeyn. Algorithms and experiments for the Webgraph. In *European Symposium on Algorithms*, 2003.

[97] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief history of the Internet. http://www.isoc.org/internet/history/brief.shtml, 2003.

[98] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion using clusters. In *28th Australian Computer Science Conference*, 2005.

[99] Chunxi Li and Changjia Chen. Gnutella: Topology dynamics on phase space, 2007.

[100] libpcap. http://sourceforge.net/projects/libpcap/.

[101] Yuting Liu, Bin Gao, Tie-Yan Liu, Ying Zhang, Zhiming Ma, Shuyuan He, and Hang Li. BrowseRank: Letting Web users vote for page importance. In *Proc. SIGIR*, pages 451–458, 2008.

[102] J. Luxenburger and G. Weikum. *Query-Log Based Authority Analysis for Web Information Search*, volume 3306 of *Lecture Notes in Computer Science*, pages 90–101. Springer Berlin / Heidelberg, 2004.

[103] F. Mathieu and M. Bouklit. The effect of the back button in a random walk: application for PageRank. In *Proc. WWW Alternate track papers & posters*, pages 370–371, 2004.

[104] A. Medina and I. Matta. BRITE: A flexible generator of Internet topologies. Technical Report BU-CS-TR-2000-005, Boston University, 2000.

[105] F. Menczer. ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. In *Proc. 14th International Conference on Machine Learning*, pages 227–235, San Francisco, CA, 1997. Morgan Kaufmann.

[106] F. Menczer. Growing and navigating the small world Web by local content. *Proc. Natl. Acad. Sci. USA*, 99(22):14014–14019, 2002.

[107] F Menczer. The evolution of document networks. *Proc. Natl. Acad. Sci. USA*, 101:5261–5265, 2004.

[108] F. Menczer. Lexical and semantic clustering by Web links. *Journal of the American Society for Information Science and Technology*, 55(14):1261–1269, 2004.

[109] F. Menczer. Mapping the semantics of web text and links. *IEEE Internet Computing*, 9(3):27–36, May/June 2005.

[110] F. Menczer and R. K. Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the Web. *Machine Learning*, 39(2–3):203–242, 2000.

[111] F. Menczer, S. Fortunato, A. Flammini, and A. Vespignani. Googlearchy or googlocracy? *IEEE Spectrum Online*, February 2006.

[112] F. Menczer, G. Pant, and P. Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology*, 4(4):378–419, 2004.

[113] Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19:395–404, 1976.

[114] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on Web usage mining. *Communications of the ACM*, 43(8):141–151, 2000.

[115] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS*, pages 50–60, 2005.

[116] David Moore, Colleen Shannon, and Jeffery Brown. Code-Red: a case study on the spread and victims of an Internet worm. In *Proceedings of the 2nd Internet Measurement Workshop*, 2002.

[117] David Moore, Geoffrey Voelker, and Stefan Savage. Inferring internet denial of service activity. In *Proc. 2001 USENIX Security Symposium*, 2001.

[118] Abbe Mowshowitz and Akira Kawaguchi. Bias on the Web. *Commun. ACM*, 45(9):56–60, 2002.

[119] Arbor Networks. Peakflow. http://www.arbor.net/products_platform.php.

[120] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, pages 167–256, 2003.

[121] M. E. J. Newman, S. Forrest, and J. Balthrop. E-mail networks and the spread of computer viruses. *Phys. Rev. E*, 66:035101, 2002.

[122] The National Laboratory for Applied Network Research (NLANR). http://moat.nlanr.net/.

[123] J. D. Noh and H. Rieger. Random walks on complex networks. *Phys. Rev. Lett.*, 92:118701, 2004.

[124] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1999.

[125] Sandeep Pandey, Sourashis Roy, Christopher Olston, Junghoo Cho, and Soumen Chakrabarti. Shuffling a stacked deck: The case for partially randomized ranking of search engine results. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proc. 31st International Conference on Very Large Databases (VLDB)*, pages 781–792, 2005.

[126] R. Pastor-Satorras, A. Vázquez, and A. Vespignani. Dynamical and correlation properties of the Internet. *Phys. Rev. Lett.*, 87:258701, 2001.

[127] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86:3200–3203, 2001.

[128] Neal Patwari, Alfred O. Hero, III, and Adam Pacholski. Manifold learning visualization of network traffic data. In *ACM SIGCOMM Workshop on Mining Network Data*, pages 191–196, 2005.

[129] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *ACM Workshop on Data Mining Applied to Security*, 2001.

[130] Feng Qiu, Zhenyu Liu, and Junghoo Cho. Analysis of user web traffic with a focus on search activities. In AnHai Doan, Frank Neven, Robert McCann, and Geert Jan Bex, editors, *Proc. 8th International Workshop on the Web and Databases (WebDB)*, pages 103–108, 2005.

[131] LLC QoSient. Argus: Audit record generation and utilization system.

[132] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing*, 6:50–57, 2002.

[133] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[134] M. Angeles Serrano, Ana Maguitman, Marian Boguna, Santo Fortunato, and Alessandro Vespignani. Decoding the structure of the WWW: A comparative analysis of Web crawls. *ACM Trans. Web*, 1(2):10, 2007.

[135] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power-laws and the AS-level Internet topology. *Transactions on Networking*, 11(4):514–524, 2003.

[136] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated worm fingerprinting. In *Proc. ACM/USENIX Symposium on Operating System Design and Implementation*, 2004.

[137] SLAC. Internet end-to-end performance monitoring. http://www-iepm.slac.stanford.edu/.

[138] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to own the Internet in your spare time, 2002. Proceedings of the 11th USENIX Security Symposium (Security '02).

[139] Marcin Sydow. Can link analysis tell us about web traffic? In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 954–955, New York, NY, USA, 2005. ACM.

[140] Jian tao Sun, Qiang Yang, and Yuchang Lu. Web-page summarization using clickthrough data. In *In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (SIGIR˘201905*, pages 194–201. ACM Press, 2005.

[141] L. Tauscher and S. Greenberg. How people revisit web pages: Empirical findings and implications for the design of history systems. *Int. J. of Human-Computer Studies*, 47(1):97–137, 1997.

[142] CERT Network Situational Awareness Team. Silk: The system for internet-level knowledge. http://tools.netsa.cert.org/silk/index.html.

[143] NWB Team. Network workbench tool. Indiana University, Northeastern University, and University of Michigan.

[144] Steve Uhlig and Olivier Bonaventure. The macroscopic behavior of Internet traffic: A comparative study. Technical Report Infonet-TR-2001-10, University of Namur, 2001.

[145] Camilo Viecco, Alex Tsow, and L. Jean Camp. Privacy-aware architecture for sharing Web histories. *IBM Systems Journal*, publication pending.

[146] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[147] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[148] Winpcap. http://winpcap.org/.

[149] Q. Yang and H. H. Zhang. Web-log mining for predictive Web caching. *IEEE Trans. on Knowledge and Data Engineering*, 15(4):1050–1053, 2003.

[150] S.-H. Yook, H. Jeong, and A.-L. Barabási. Modeling the Internet's large-scale topology. *PNAS*, 99:13382–13386, 2002.

[151] Yin Zhang, Sumeet Singh, Subhabrata Sen, Nick Duffield, and Carsten Lund. Online identification of hierarchical heavy hitters: Algorithms, evaluation, and applications. In *Internet Measurement Conference*, pages 101–114, 2004.

[152] Cliff Zou, Don Towsley, and Weibo Gong. Email worm modeling and defense. In *Proc. 13th International Conference on Computer Communications and Networks (ICCCN'04)*, 2004.

MARK MEISS

Advanced Network Management Laboratory
mmeiss@indiana.edu
Innovation Center, Suite 201 http://steinbeck.ucs.indiana.edu/~mmeiss
Indiana University Voice +1-812-855-1878
Bloomington, IN 47403, USA

Research interests

Network measurement, Web search and data mining, traffic modeling, network security, anomaly detection, graph analysis, machine learning, information retrieval, complex networks and systems, behavioral analysis, social networks, distributed computation, network protocol design and analysis.

Academic history

Ph.D. Candidate (ABD)	*Computer Science*, Indiana University, Bloomington, 2006 (Anticipated completion in May 2010.)
B.S.	*Computer Science*, Indiana University, Bloomington, 1999
B.S.	*Mathematics*, Indiana University, Bloomington, 1998

Professional history

2008-2009	Instructor, School of Informatics, Indiana University
2001-present	Researcher, Advanced Network Management Laboratory, Pervasive Technology Labs, Indiana University
1997.2001	Senior Network Manager and Unix Specialist, University Information Technology Services, Indiana University
1996.1997	Programmer/Editor, University Information Technology Services Knowledge Base, Indiana University
1995.1998	Programmer, Sabbagh Associates, Bloomington, Indiana
1993-1995	Programmer, Department of Ophthalmology, Indiana University School of Medicine

Personal data

Date of birth:	February 26, 1976
Citizenship:	USA
Permanent residence:	USA

Grants and fellowships

2002-2005	NSF Grant "Network Management Tools for End-to-End Performance Measurement" (Senior Personnel)
2001	Indiana University Graduate Fellowship
1997-1999	Goldwater Fellowship

Awards and honors

2006 Iota Nu Phi (Alpha Chapter, Informatics Honor Society)
1998 Phi Beta Kappa
1995-1998 Indiana University Metz Scholar
1995 Golden Key Honor Society
1994-1995 Indiana University Reynolds Scholar
1994 National Merit Scholar Semi-Finalist

Professional service

- Program committees:
  ACM Conference on Hypertext and Hypermedia (HT 2010)
  ACM Conference on Hypertext and Hypermedia (HT 2009)

- Conferences refereed**:** *AAAI, WWW, WSDM*

Teaching experience

U.G. courses (IU)
INFO I211 Information Infrastructure II (F2008, F2009)
CSCI P438 Fundamentals of Computer Networks (Invited Lectures) (F2004)
CSCI P442 Digital Systems (Lab) (S1998)

Programming Experience

- Extremely fluent in C, Perl, Python, Java, Processing, and Scheme.
- Somewhat experienced with MATLAB, LabVIEW, SQL, Javascript, C++, Objective-C, Smalltalk, PHP, assembly language (6502, 6809, 68000, PIC, AVR), Visual BASIC, Postscript, elisp, and Pascal.
- System administration experience on Linux, FreeBSD, and Solaris.
- Web development experience with HTML, CSS, XML, XSLT, and AJAX.

Publications

A. Journals

[A.4] M. **Meiss**, F. Menczer, A. Vespignani: Properties and evolution of Internet traffic networks from anonymized flow data. *ACM Transactions on Internet Technology*, under review.

[A.3] M. **Meiss**, F. Menczer: Visual comparison of search results: A censorship case study. *First Monday* 13(7), 7 July 2008.

[A.2] M. **Meiss**, F. Menczer, A. Vespignani: Structural analysis of behavioral networks from the Internet. *J. Phys. A: Math. Theor.* 41: 224022, 2008.

[A.1] K. Börner, S. Penumarthy, M. **Meiss**, W. Ke: Mapping the diffusion of scholarly knowledge among major U.S. research institutions. *Scientometrics* 68(3): 415-426, 2006.

B. Peer reviewed conferences and workshops

[B.8] M. **Meiss**, B. Gonçalves, J. Ramasco, A. Flammini, F. Menzcer: Agents, Bookmarks, and Clicks: A topical model of Web navigation. Submitted to ACM Conference on Hypertext and Hypermedia (HT), under review.

[B.7] M. **Meiss**, J. Duncan, B. Gonçalves, J. Ramasco, F. Menczer: What's in a session: tracking individual behavior on the Web. Proc. 20th ACM Conference on Hypertext and Hypermedia (HT), 2009.

[B.6] M. **Meiss**: An analysis of sampling effects on graph structures derived from network flow data. FloCon, 2009.

[B.5] B. Gonçalves, M. **Meiss**, J. Ramasco, A. Flammini, F. Menczer: Remembering what we like: Toward an agent-based model of Web traffic. Late-breaking result at 2nd ACM International Conference on Web Search and Data Mining (WSDM), 2009.

[B.4] M. **Meiss**, F. Menczer, S. Fortunato, A. Flammini, A. Vespignani: Ranking Web site with real user traffic. Proc. First ACM International Conference on Web Search and Data Mining (WSDM), pp. 65-78, 2008. [Acceptance rate 16%]

[B.3] M. **Meiss**, F. Menczer, A. Vespignani: Building networks from networks: mining network data to model user behavior. IPAM Workshop on Dynamic Searches and Knowledge Building, UCLA, 2007.

[B.2] M. **Meiss**, F. Menczer, A. Vespignani: A framework for analysis of anonymized network flow data. Proc. National Science Foundation Symposium on Next Generation of Data Mining and Cyber-Enabled Discovery for Innovation (NGDM), 2007.

[B.1] M. **Meiss**, F. Menczer, A. Vespignani: On the lack of typical behavior in the global Web traffic network. Proc. 14th Intl. WWW Conference, pp. 510-518, 2005. [Acceptance rate 14%]

C. Book chapters and contributions

[C.1] M. **Meiss**: Race Pharming. *Phishing and Counter-Measures: Understanding the Increasing Problem of Electronic Theft and Identity*, edited by M. Jakobsson and S. Myers. Wiley, 2007.

Software creations and online resources

- Abilene Weather Map: a real-time map of network traffic on the Internet2 data network. (Replaced by newer version in 2007.)
- cenSEARCHip: a tool to visualize the differences in the results returned by different countries' version of the major search engines. Covered in various news outlets around the world.
  http://homer.informatics.indiana.edu/censearchip/
- Flow Generation Language (FGL): a programming language for automated generation of network test traffic among many endpoints.
- FlowSeek: an optimized query system for analysis of network flow data.
- Loner: a software system for probabilistic location of faulty network connections in a campus environment.
- Tsunami: a high-speed UDP-based file transfer protocol for next-generation data networks. Used in Internet Land Speed Record efforts in 2002-2003.
  http://anml.iu.edu/research.shtml
- Very Large Graph Reducer (VLGR): a software system for sampling and reducing graphs for other analysis.

Invited external talks and contributions

2007 Invited speaker, IPAM Workshop on Dynamic Searches and Knowledge Building, UCLA, California.
2006 Participant, Winter School on Modeling and Mining of Network Information Spaces. Banff, Alberta.
2006 Invited speaker, Complex Systems Network Seminar, Bloomington, Indiana.
2006 Invited speaker, I690: Applied Cryptography, Indiana University School of Informatics, Bloomington, Indiana.
2005 Presentation, Internet2 Members Meeting, Columbus, Ohio.
2004 Organizer and Speaker, Advanced Network Management Laboratory Security Workshop, Bloomington, Indiana.
2003 Presentation, Internet2 Members Meeting, Tempe, Arizona.