UNIVERSITY OF SOUTHERN QUEENSLAND
AUSTRALIA

Estimation and application of social network behaviour from network traffic

A Thesis submitted by

Mostfa Mohsin Albdair

For the award of

Doctor of Philosophy

2018

Abstract

Because traffic is predominantly formed by communication between users or between users and servers which communicate with users, network traffic inherently exhibits social networking behaviour. In this thesis the extent of interaction between entities – as identified by their IP addresses – is extracted from massive anonymized Internet trace datasets obtained from the Center for Applied Internet Data Analysis (CAIDA) and analysed in a multiplicity of ways. A key discovery is that the Pareto principle applies to all the key features which have been identified: sources of traffic, destinations of traffic, traffic flows themselves, and also to the abstract sources, destinations, and flows which are identified by spectral analysis of traffic matrices.

Chapter 2 reviews the literature on traffic analysis emphasising social network behaviour, and also the literature on methods for managing Quality of Service in the Internet. In Chapter 3, we infer the behaviour of social networks from O–D pair traffic by using two groups of analysis methods. The first group of methods is based on frequency analysis. The frequency analysis of origin, destination and O-D Pair statistics are found to exhibit heavy tailed behaviour. The second method is a slightly different characterization of social behaviour.

In Chapter 4, traffic matrices are first decomposed into mean and variation about the mean. The mean traffic matrix is then analysed by singular value decomposition. The variation about the mean is analysed separately in two different ways. First, assuming that traffic associated with one O-D pair is uncorrelated with traffic of any other O–D pair, the variation is analysed by singular value decomposition. Secondly, dispensing with the O–D-traffic independence assumption, the covariance matrix of O–D traffic is analysed by singular value decomposition. It is discovered that in all three cases, the Pareto principal, that a small proportion of eigenvalues explains a high proportion of the matrices, is found to be true. The significance of the covariance eigenflows was tested by simulating a network with independently distributed O-D traffics of on-off type, with Pareto distributed on and off periods. The simulation confirmed the significance of the covariance eigenflows. A new software system comprising more than 7,000 lines of C++ code for analysis of very large pcap trace files, called *Antraff*, has been developed to carry out all the analysis procedures in Chapters 3 and 4.

In Chapter 5, the understanding of the social–network behaviour exhibited by traffic is applied to design traffic control procedures which have highly significant advantages for maintaining QoS in the Internet. An architecture for protecting QoS is introduced, based on the understanding of social behaviour exhibited by traffic. Whereas DiffServ enables different treatment to be given to different traffic types, in this architecture, known as DefServ, different treatment is given on the basis of the traffic situation.

Certification of Thesis

This Thesis is entirely the work of **Mostfa Mohsin Albdair** except where otherwise acknowledged. The work is original and has not previously been submitted for any other award, except where acknowledged.

Student and supervisors signatures of endorsement are held at the University.

Principal Supervisor: Assoc Prof Ron Addie ________________ Date: ______________

Associate Supervisor: Assoc Prof Alexander Kist ________________ Date: __________

Associate Supervisor: Dr. Shahab Abdulla ________________ Date: ________________

Acknowledgments

In the name of Allah, All-Merciful

and

The blessings and peace of Allah be upon our Master Muhammad, his family and his Companions.

“If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas.”

— George Bernard Shaw

On my irregular but worthwhile journey toward my PhD degree. Over the past five years I met not only challenges in work and life, but also many supportive individuals who gave me the confidence to overcome those challenges. I take this opportunity to thank everyone who gave me their valuable assistance during my Ph.D. study at University of Southern Queensland for their remarkable guidance and help. Without these people, the establish of this task would have been more difficult.

First and foremost, I would like to express my greatest gratitude to my supervisor Assoc Prof Ron Addie for his support, knowledge, patient and belief in me. I will always consider myself most fortunate to have had the opportunity to work under the supervision of Assoc Prof Ron Addie. He made my PhD study a great experience, as much scientifically than personally. I would like to thank my second supervisor Assoc Prof Alexander Kist for sharing his knowledge and providing insightful comments in the research work. Also, I would like to thank my third supervisor Dr. Shahab Abdulla for his assistance and suggestions in the preparation of this thesis.

Also, I would like to thank Center for Applied Internet Data Analysis (CAIDA) for providing the Internet traces datasets in this research.

I would like to take this opportunity to thank my country, and the government of Iraq represented by the Ministry of Higher Education and Scientific Research for supporting me and my family with scholarship.

The financial supports from the Australian Commonwealth Government contribution through the Research Training Program (RTP) Fees Offset scheme during my research is much appreciated also.

Living and studying in Australia for my PhD has been an amazing journey, not because of all the miles flown, but because I met some great people along the way.

Being so far from home means that I did not see my family as much as I would have wished. I thank them all for their support and understanding.

Mostfa Mohsin Albdair

Associated Publications

M. Albdair, R. G. Addie, and D. Fatseas, (2017), 'Social network behaviour inferred from O–D pair traffic', Australian Journal of Telecommunications and the Digital Economy 5(2), 131–150.

R. G. Addie, Y. Peng, M. Albdair, C. Xing, D. Fatseas and M. Zukerman, "Cost modelling and validation in network optimization", in Proc. International Telecommunication Networks and Applications Conference 2015, pp.11-16, Sydney NSW, Nov. 2015.

M. Albdair, R. G. Addie, and D. Fatseas, "Inference of social network behavior from internet traffic traces," in Telecommunication Networks and Applications Conference (ITNAC), 2016 26th International. IEEE, 2016, pp. 7–12.

Contents

List of Figures

List of Tables

List of Listings

Acronyms & Abbreviations

IETF	Internet Engineering Task Force
QoS	Quality of Service
O-D pair	Origin-Destination pair
TM	Traffic Matrix
CAIDA	Center for Applied Internet Data Analysis
PCA	Principal Component Analysis
DiffServ	Differentiated Services
DES	Discrete Event Simulation
RSVP	The Resource Reservation Protocol
MPLS	Multiprotocol Label Switching
LTE	Long Term Evolution Protocol
TCP	Transmission Control Protocol
IOS	iPhone operating system
IP	Internet Protocol
IPv4	Internet Protocol version 4
RWNDQ	Receiver window queue
SVD	Singular-value decomposition
DSCP	Differentiated Services Code Point
SJF	shortest-job-first
BE	Best Effort
Defserv	Defensive Services
MCR	Multi-hop Communication Routing
MAE	Mice and Elephants Queue Discipline
pFifoFast	priority first input first output Fast Queue Discipline
TOS	Type Of Service

Chapter 1

Introduction

Both the traffic and the technology of networks is dramatically different from the past. As the technology expands the power and flexibility of our communication pathways, the ways in which networks are used adapts and even begins to suggest different pathways for future evolution.

In recent years, the use of Internet applications and the variety of these applications has grown enormously. According to (**?**), in 2016 the Internet traffic has reached up to 10 GB per capita, and in 2021 it will reach 30 GB per capita. In 2016, global IP traffic was 1.2 ZB per year and it will be about 3.3 ZB per year in 2021. Also, the volume of the entire Internet traffic will increase to 127 times between 2005-2021 (**?**). For this reason, it is recognised that we need to collect and analyse the Internet traffic which is obtained, in this thesis, from the the Center for Applied Internet Data Analysis (**?**).

Analysing and understanding Internet traffic is perceived by some researchers as crucial for understanding the performance experienced by Internet users. Furthermore, it is clear that traffic must logically be viewed as both the reflection of, and a critical insight into the social network formed by Internet users and its services. Furthermore, the huge amount of information which is passing through the Internet networks makes this task very difficult. The Internet is moving from providing only best effort services to classifying traffic to support different Quality of Service (QoS) for different flows.

Attempting to improve and refine management of operational performance of the Internet can be undertaken on a more scientifically sound basis, and with greater confidence of success, if we we have a well understood approach for analysing network traffic, or, even better, a model of network traffic which can be relied on to explain it.

1.1 Social Networks

Concepts from the theory of social networks have been used in many fields; kinship structure, social mobility and science citation (?), epidemiology (?) and (?), politics (?), economics geography (?), education (?), psychology (?), information sciences (?), communication technology (?) and many other fields. According to (?), social network theory provides a powerful model for social structure. The author also claimed that a much broader meaning of social network has been used over an extended period in sociology.

A variety of methods have been suggested to examine and interpret the relationships of the participants of a community or group into a system and this is what has been called *social network analysis* (?). In (?), social network analysis attempts to understand the members of the networks and the relationships between them. Social network analysis assumes that the relationships of the members are important in terms of understanding the behaviour of the participants, i.e. who knows whom, and who shares what information and knowledge with whom by what communication media. Research into social networks preceded and informed the development of Internet applications such as facebook, twitter, linkedIn, ResearchGate, Academia, SnapChat, and Instagram. Also, apps such as skype, WhatsApp, Facetime, Google Duo, and WeeChat which are primarily designed to provide point-to-point or group communication make use of the social network framework to present functionality in a more convenient form. These products are now so pervasive that the term "social network" is often conflated with using these applications. Indeed, social-networking applications are sufficiently important now that research into their history, development, and use is a field of study in its own right.

In this thesis, however, the concept of social network refers to the underlying relationships between people as evidenced by the volume and statistics of traffic between them. Whereas usually in social network analysis, the relationships are modeled by the *links* in the network, in this work, the relationships are modeled and measured by the *traffic* between members. The study of *traffic* in networks began decades before social network theory began (**?**). This conference was founded back in 1955 and it has established a multi-decade tradition as the primary forum for presenting and discussing the latest technical advances in the broad areas of teletraffic models, network systems, and measurements. An established way to study social networks is by modelling the structure of their communication networks (**?**). A communication network is usually understood to be made up of two distinct types of components: nodes (or vertices), and links (or edges). Thus, the term network seems to be synonymous, at its core, with the mathematical concept of a *graph* (**?**). In the theory of graphs, the two key types of object are known as vertices and edges. According to (**?**), graph theory has an extensive literature, and many applications, of its own. The key distinguishing feature of networks from graphs is probably the fact that we assume, generally, that networks are *used* by their nodes to provide end-to-end communication (or some activity like communication, e.g. the passing of an infection). In communication networks, the activity which takes place, to use a network for communication, is known as *traffic*.

Social network theory in a simplistic sense is an application of the theory of communication networks to social groups. The theory of social networks has now developed sufficiently that its techniques can be usefully applied to communication networks (from where the first of its techniques were derived). However, in this process of applying social network theory to its original source, in the field of communication networks, it is important not to neglect the dual structure of communication networks as comprising nodes, links, *and traffic* (**?**) and (**?**).

Although operators have access to traffic data from within their own networks, sharing this traffic data is rare, because of privacy and commercial considerations. Large quantities of traffic data from public networks has, however, been collected and is publicly available (**?**). Analysis of this data, and its implications, are discussed in Chapters 3 and 4. In this thesis we study the behaviour of social network traffic by analysing massive datasets of Internet traffic obtained from CAIDA using a new software package (**?**), which undertakes statistical analysis of traffic data in a variety of ways. Although network traffic has been studied extensively, looking at traffic as representation of the behaviour of the whole network, rather than as a representation of the behaviour of one link has rarely been considered.

1.2 The Pareto Principle

According to (**?**), more than a hundred years ago the Italian economist and sociologist Vilfredo Pareto discovered the principle that 20% of the population owned 80% of the property in Italy. After that, Pareto created a mathematical formula to describe the unequal distribution of wealth in his country, which is known as the Pareto distribution.

The Pareto principle arises in human behaviour in many disciplines; business to improve productivity and make better decisions (?), health (?), education (?), psychology (?) and (?), politics (?) and many other areas.

The Pareto principle was first used in traffic models in (?). Research into the mathematical analysis of such traffic models was very active subsequently (?), (?), (?) and (?). Although the details of the models varied, the primary interest was always to predict the performance experienced on a link.

The Pareto principle for characterising traffic emerged in the study of *measurements* of traffic on a link in (?) and later measurements confirmed the existence of Pareto distributed flows in measurements of link traffic (?), (?), (?) and (?). In (?), the Poisson Pareto burst process (PPBP) was suggested and examined as as a model for Internet traffic. This model was predictable for the future multiplexing and link efficiency levels. The measurements of traffic on the links of a network is formulated and discussed under Poisson assumptions in (?). An alternative name sometimes used for the Pareto distribution is *power law distribution.* Another characterization of systems with Pareto distributed features is that they exhibit self-similarity (?), or asymptotic self-similarity. They argue that the Pareto distribution has been shown to be useful for modeling VoIP, IP and Ethernet amongst other applications. According to (?), the Pareto distribution, as shown in many empirical studies, provides a good fit for a number of important characteristics of communication networks traffic. According to (?), data communication of packets follows heavy tailed distributions in the Internet network. The authors of this paper examined characteristics of IPv6 and IPv4 traffic obtained from MAWI working group traffic archive.

1.3 Modelling whole network traffic

There are two kind of network traffic models; traffic measured on a single link and methods that apply to traffic measured simultaneously on all links of a network. A model of Web traffic, for example, that attempts to capture the major aspects of the traffic has been presented in (?). In (?), bipartite graphs and one-mode projection graphs have been used for modeling data communication in network traffic. In (?) different Internet traces have been examined based on Poisson assumption, the authors found that the traffic seems to take on a significantly non-stationary form of behaviour.

The behaviour of Internet traffic must be correctly understood in order to design robust computer networks and network services with better quality (?). A whole network traffic model will enable us to see a clear picture for the behaviour of traffic which is statistically sound. Whole network traffic analysis is difficult because the number of origins, destinations, and origin-destination pairs which arise may be so large that naive methods of analysis may fail, or the time taken to produce a result may take too long (?). In some cases there may be more than 1 million distinct sources, destinations, or O-D pairs referenced in a sample. On the other hand, in our analysis it may be necessary to identify correlations *between* different origins, or destinations, or O-D pairs, which will not be possible without some carefully designed algorithms.

To gain a better understanding of whole-of-network traffic it is useful to focus on origin-destination flows (O-D flows) (?). We need to compare and contrast different origins, different destinations, and different O-D flows. This analysis needs to be statistical in its framework, and if possible we would like to identify the key statistical features which explain network behaviour in terms of origins, destinations, and O-D flows. Traffic traces captured in an access network will, obviously, provide information mainly about the nodes in this part of the network, and the flows to and from those nodes. Traces from links in the core network, however, will provide information about a large proportion of the activity in the network. A relatively small proportion of core networks links can be expected to cover a large proportion of flows and nodes in the whole network. Furthermore, the *statistical features* observed in even one core network link can be expected to quite similar, whichever core link is used to collect the trace.

The traffic trace from one core link will, as we shall see in Chapters 3 and 4, reveal the presence of large flows. These large flows will, however, not necessarily cause congestion on the link where the observations take place. This is because, although most large flows will cause congestion *somewhere* in the network, the location of the congestion will vary from case to case. In general, it is much more likely that the congestion will occur in some part of the access network. Thus, the critical information from a core network traffic trace is likely to convey important information, which is relevant to the traffic management which needs to take place in a variety of locations in the network.

In this thesis, the Pareto principle has been observed in the traces of Internet traffic in multiple ways – in sources, destinations, and O-D pairs, as well as in a more statistical sense when spectral analysis of traffic matrices is undertaken – as found in Chapters 3 and 4. The fact that Pareto behaviour, which has been observed by researchers in almost every field where human behaviour is a consideration, appears in such a diverse and intrinsic way suggests strongly that the Pareto behaviour in Internet traffic is not an artifact, but reflects the needs and patterns of behaviour of Internet users. The consequences of this observation are far-reaching. In particular, it suggests strongly that the Pareto feature of Internet traffic is likely to remain important for a considerable period of time, even as applications emerge, rise in importance, and are then replaced by new applications.

1.4 Improving Service

As the Internet becomes more important in society, it becomes more important to understand network traffic behaviour and provide sufficient QoS (**?**). As the number of users increases, also the rate of usage of each user increases, the range of different *types* of use of each user increases, and new types of use emerge. As communication through the Internet becomes more important in the way we work and live, the requirement to achieve satisfactory quality of service (QoS) becomes crucial. Several large–scale evaluations on real–world web services have been conducted to investigate QoS (**?**).

The conflicting requirements of mice and elephant flows should be considered (**?**). They proposed the RWNDQ mechanism to feedback queue occupancy levels to TCP senders. Providing better performance for Internet traffic, especially for traffic formed of short–lived flows 'mice', based on the behaviour of traffic is important (**?**). According to (**?**), the performance of short flows 'mice' which is the majority of TCP flows is particularly important.

In the design of Internet architectures, the modelling of Internet traffic is represented as critical task. Many studies have focused on modelling source traffic related to specific application-level protocols, for the purpose of conducting realistic network traffic simulation experiments (**?**). In this thesis, in Chapter 5, we will design and test a new method to improve the quality of service called Defensive Services 'Defserv'. This method is based on a new mice and elephants queue discipline (MAE).

1.5 Philosophy and aims of this thesis

Estimation of social network behaviour from traffic is a highly complex undertaking with potential implications for a wide range of research topics, such as Internet traffic architecture, routing policies or link dimensioning. In this thesis we narrow the scope of our work to the estimation and application of social network behaviour from network traffic traces. The implications of the behaviour discovered in this way are then investigated in Chapter 5.

1.6 Contributions

This thesis makes the following main contributions:

- Identification of new statistics features reflecting underlying social network behaviour from large traffic traces, more specifically:
  - Probability model of Source activity;
  - Probability model of Destination activity;
  - Probability model of O–D pair activity.
- Introduction of a new three-stage spectral analysis of traffic matrices, first decomposed into mean, variance and covariance matrices, which are then further analysed by singular value decomposition.
- Discovery of evidence of correlation of flows, by comparison of spectral analysis of covariance vs spectral analysis of variance matrices. Presence of correlation of flows is then further tested by comparison with simulations in which correlation is not present.

- Identification of Pareto behaviour of Source traffic, of Destination traffic, and of O-D pair traffic.
- Estimation of the Pareto shape parameters of the eigenflows (or eigensources or eigendestinations).
- Development of simulations of traffic with Pareto flow sizes and estimation of the throughput vs flowsize response, from these simulations.
- Introduction of DefServ architecture as an alternative to DiffServ, based on responding to traffic status, rather than based on pre-existing statically defined classes of traffic.
- Development of a new queue-discipline (MAE) in the ns-3 software.
- Introduction of a new way to analyse and display queueing performance (plots of flow-size vs net throughput, for flows, with least-squares fitted linear trend).
- Comparison of a traffic control strategy based on the Pareto character of underlying traffic and demonstration of its performance advantages relative to DiffServ and Droptail.
- Use of the Pareto shape parameter estimated from traffic traces to estimate the frequency with which overload events will normally occur.
- Observation and checking that the Pareto behaviour of traffic implies that the operation of the new traffic control strategy should be robust and efficient (since the only requirement is to detect elephants, which are statistically robust to observe).

1.7 Organization of the thesis

This thesis consists of six chapters, with the current chapter presenting an introduction to the thesis. The remaining chapters of the thesis are structured as follows:

- *Chapter* 2 summarizes the related work in the social network traffic modeling and quality of services enhancement. Techniques used in the analysis of network traffic collected from are discussed along with their challenges and limitations. A detailed review of different techniques used in building the Internet traffic models is also provided in this chapter.
- *Chapter* 3 identifies and estimates the first-order characteristics (particularly the byte-frequencies per source, per destination, and per O-D pair) of the underlying social network from the Internet trace datasets obtained from the Center for Applied Internet Data Analysis (CAIDA). The frequency statistics of origin, destination and O-D Pairs exhibits heavy tailed behaviour (the Pareto principle). The Pareto shape parameters for these separate cases are estimated.
- *Chapter* 4 undertakes spectral analysis of traffic matrices estimated from the CAIDA traces. Traffic matrices of three different types are considered: mean, variance, and covariance matrices. A discrepancy between the eigenvalues found in the singular-value decomposition of the covariance matrix from those found in the mean and variance matrices suggests the existence of correlations between O-D flows. Simulation experiments are conducted to confirm the hypothesis that O-D flows are correlated.
- *Chapter* 5 a new approach to protect quality of service which uses the Diffserv architecture, but which concentrates on defense rather than protection of quality of service will be defined, analysed and compared.
- Finally conclusions and future research directions are provided in *Chapter* 6.

Chapter 2

Background

2.1 Internet Traffic Modeling

The authors of (?) and (?) reported that the design of robust and reliable networks and network services is becoming increasingly difficult in today's world. Given the impact of the increasing in the size of Internet traffic, it is clear that the analysis of huge traffic is of critical importance. One way to assist this goal is to develop a detailed understanding of the traffic characteristics of the network.

In this thesis a mathematical model of traffic will be developed by identifying the key features of this traffic by statistical analysis of traffic traces. One of the most prominent methods for identifying features of any system is principal component analysis (PCA). Not all the methods used in this dissertation are based on principal component analysis, however these methods do form a key focal point. According to (?), analyzing massive amount of Internet traffic is crucial issue and it is as yet unmet. In this dissertation, methods for analysing traffic generated by tens of millions of O-D pairs are developed by first identifying the origin IP addresses, destination IP addresses, and O-D pairs which occur with more than marginal frequency. This is one of the tasks achieved in Chapter 3.

Quality of Service (QoS) management is an issue of on-going concern which can also benefit from accurate understanding of traffic statistics. For current and future networks, modelling of network traffic is significant to understand and solve the problems of networks performance. A number of the traffic models have been suggested in previous studies. For instance, a web traffic model is one of these models which is referring to the data that is sent or received by the browser of user websites (?). Internet traffic models are useful for the development of telecommunication technologies, in order to analyse the performance and capacity of various protocols, algorithms and network topologies.

Given the importance of modelling the traffic matrix (TM) for number of network aspects, good models of traffic matrices are important. (?) expressed concerned with the paucity of studies which are focused on complete models for TM, despite the importance of TM modelling. Also, (?) state that the number of methods which are proposed in the literature to address the TM estimation problem in the context of Internet traffic is very small. Traffic modelling and analysis is an essential preliminary step in network design and management. A number of point-to-point traffic models have been studied and discussed in the literature. For example, (?) formulated and discussed the estimation of node-to-node traffic on the links of a network under Poisson assumptions. Also, (?) proposed optimization of QoS over multiple domains supporting Software Defined Networking.

Furthermore, (?) and (?) analyse the traffic behaviour of individual hosts and applications. While the clustering algorithms on the similarity matrices of one–mode projection graphs have been applied in (?) to study the behaviour clusters of social networks. The authors of this paper found significant similarity of social behaviour among different Internet applications. However, there is a paucity of work currently available on models of the entire traffic of a network.

In order to develop QoS architecture within the Internet networks, a strong traffic model is required for accurate performance evaluation and traffic analyses (?). What is needed is not only a model for the traffic on each link, but a model for the traffic of the whole network (?). This is the type of model we seek in this thesis.

In (?) and (?), the authors acknowledge that an analysis of real traffic is important to understanding better the nature of the traffic in terms of the underlying distributions, correlations, power laws etc. Such information could help us to build realistic theoretical traffic models. This means that traffic analysis can provide a lot information about the network such as average load, the bandwidth requirements for different applications, and many other details. In this thesis the statistical analysis of Internet traffic has been used to improve service management. In Chapter 5 we introduce a new approach to provide better Quality of Services from social network behaviour understanding called Defensive Services 'Defserv'.

In previous studies, power law traffic models have been identified as suitable for the flows on one link. However, in this thesis, what is sought is a model for the traffic of the whole network. A whole network traffic model will enable analysis of the performance of alternative architectures which is statistically sound. For example, with a whole network traffic model, simulations can be conducted which range over the situations which might be expected in a real Internet, in a statistically valid way. Using a small collection of scenarios selected arbitrarily cannot provide evidence for how well one architecture performs relative to another, whereas a collection of scenarios drawn from a carefully estimated model of network traffic can. Network designers can use the traffic model to make better prediction about Internet traffic behaviour. According to (?), the availability of realistic traffic models is required to improve the performance of the current and future networks. For these reasons, the task of developing a whole-of-network Internet traffic model could be critical.

In (?), the authors highlighted that the traffic model in any network services or architectures is very important. The traffic model is the heart of the performance evaluation of telecommunications networks. The precise estimation of network performance is the avenue for the success of any service. For example, Markov processes and regression models are used to model the short–range dependent traffic. The networks need to assure an acceptable level of QoS to the users. Therefore, traffic models need to be accurate and able to capture the statistical characteristics of the actual traffic. (?) stated that there are two fundamental ways which use the traffic models. These ways are (1) as part of an analytical model or (2) to drive a Discrete Event Simulation (DES).

According to (?) the performance of models is important in terms of deciding the level of QoS for networks. The authors of this paper have achieved the improvement of the network performance by implementing two network optimizations mechanisms. Accurate modeling of performance requires traffic models with high accuracy. This implies that traffic models must be able to capture the statistical characteristics of the actual traffic on the network. If the underlying traffic models do not efficiently capture the characteristics of the actual traffic, the result may be the under-estimation or over-estimation of the performance of the network or, what is potentially even worse, misinformed choice of architecture and traffic control strategies. In particular, it is discovered in this research, that a Pareto principal applies to traffic in the whole network, not just to the flows on an isolated link.

2.1.1 The Pareto Principle

According to (?), more than a hundred years ago the Italian economist and sociologist Vilfredo Pareto discovered the principle that 20% of the population owned 80% of the property in Italy. After that, Pareto created a mathematical formula to describe the unequal distribution of wealth in his country, which is known as the Pareto distribution. It is often used in management, economics and business to improve productivity and make better decisions. Later on, it was discovered that the same principle can be applied to Internet traffic (?). Recent studies have also found that Internet traffic exhibits Pareto principle (??).

This model is an example of a power law distribution which has been used to model the self similar characteristics of applications (?). They argue that the Pareto distribution has been shown to be useful for modeling VoIP, IP and Ethernet amongst other applications. According to (?), the Pareto distribution, as shown in many empirical studies, provides a good fit for a number of important characteristics of communication networks traffic. In this thesis many traffic characteristics are inferred from Internet traffic understanding to get clear picture for traffic behaviour.

2.1.2 Principal Component Analysis

Principal component analysis (PCA) is one of the best known tools in statistics and data analysis more generally. Many techniques in data mining are based on PCA or the Singular Value Decomposition Scalable Video Coding (SVD) (?).

PCA identifies a series of *features*, and provides a numerical measure of their relative importance in the object or system being analysed. The numerical measure, which is an eigenvalue of a decomposition of a matrix, can be interpreted as the *power* (or variance), of the particular feature. The features with the largest eigenvalues might be regarded as anomalies, or merely as problem traffic, which should be controlled. Understanding the behaviour of such traffic may be very important (?), (?). In other words, PCA provides an enumeration of traffic features, each of which is identified by the corresponding eigenvector, and these features will therefore be potentially controllable.

An example of problem traffic is an Apple IOS update. It has a specific origin, and as "destination" it has a *set of destinations.* This type of traffic is normal, occurs rarely, but is very important when it occurs. On each occurrence, of this particular type of event, the origin will be the same, but the destinations will be random.

The traffic model, in this thesis, includes a list of such types of events, where the randomness can be associated with the origin, or destination. The size of the event will be random. Once problem features have been discovered, monitoring them for the purpose of dis–privileging this traffic as required can be initiated. This model is used in Chapter 5 to develop a traffic control architecture. According to (?), various distributions, including Pareto, have been used to model the self similar characteristics of applications. They argue that the Pareto distribution has been shown to be useful for modeling VoIP, IP and Ethernet amongst other applications. According to (?), the Pareto distribution, as shown in many empirical studies, provides a good fit for a number of important characteristics of communication networks traffic. In this thesis many traffic characteristics are explored in order to develop a better overall picture of traffic behaviour.

In Chapter 3, Pareto models of the volume of traffic associated with sources, destinations and with O-D pairs are developed, and then, based on this foundation, in Chapter 4 a traffic model is developed by Principal Component Analysis (PCA) method and, more generally, singular value decomposition, of end-to-end traffic as a whole. The key parameters in traffic data for PCA are source address, destination address, time and data length. Most existing work using PCA in application to networks focuses on analysis of traffic to discover anomalies. However, the same approach should be applicable to analyse traffic with the new interpretation as a traffic model fitting procedure (?) and (?).

2.1.3 Gravity Models

Gravity models have been used to model flows of transport, population or trade in spatial networks (ie: in a geographic sense) (?). The volume of interaction T_{ab} between two locations (ie: trade flows between Town A of a Size N_a and Town B of a Size N_b separated by a distance d can be expressed in terms of the gravity Model (?) as follows:

$$T_{ab} = N_a.N_b.f(d_{ab}).$$

(?), claim that the gravity model is one of the most popular models in connection with traffic matrices, which estimates the Origin-Destination (OD) flow counts from ingress and egress counts. This model has been used extensively in traffic matrix estimation.

These gravity Models resemble Newton's Universal Law of Gravitation where the force (ie: volume or extent of interaction, in our case) is directly proportional to the product of the nodes' mass (ie: the degree or level of importance of the node) and is inversely proportional to the distance that separates these two nodes. Now considering the extent of interactions T_{ab} between two "nodes" in the Internet, this can also be expressed using this same gravity Model (?) where N_a provides the level of importance of source a (ie: number of unique source IP Addresses for a particular node); N_b provides the level of importance of destination b (ie: number of unique destination IP Addresses for a particular node), and where f is the deterrence function (ie: describes the influence of distance/cost between the source and destination nodes or sometimes referred to as the distance decay function).

If the gravity Model applies, then the deterrence function f will exhibit scale free properties and can be expressed in terms of a power law (ie; due to the distance/cost between two nodes or two locations being excessively high, then the degree of connectivity and likelihood of hubs existing between these two locations will be low). That is, the deterrence function f and its display of scale free properties can expressed in terms of a power law:

$$f \sim d_{ab}^{g},$$

where g is an index.

In this thesis, in Chapter 4, an alternative approach to modeling sources and destinations of traffic in a social network is developed. The concept of eigensource and eigendestination is inferred from the Singular–value decomposition (SVD) of traffic matrices. A natural interpretation of this concept is that, eigendestination is like a club, e.g Youtube users, with many users included in each eigendestination. A similar interpretation applies to eigensources, i.e. an eigensources is also like a club. A service like Youtube is not necessarily provided by a single server. The major servers, like Youtube, are not necessarily an individual server but may be a group of servers.

2.2 QoS Management

Quality of Service (QoS) is the capability of a network to provide service with a planned level of performance (as regards throughput, delay, and packet loss) to selected network traffic based on their preference. In the next generation of Internet, Quality of Service (QoS) requirements are raised by a wide range of communication-intensive, real-time multimedia applications. Also, with the development of Internet services, the problem of defining mechanisms to guarantee the transmission parameters became an important issue. Therefore, the Internet Engineering Task Force (IETF) has proposed service models and architectures to provide better services and to meet the demand for QoS. Notably among them are the Differentiated Services (DiffServ) model (?) and (?), the Integrated Services/RSVP model (?) and (?), MPLS (?), Traffic Engineering (?) and Constraint Based Routing (?).

The Differentiated Services architecture (DiffServ) is one of the most prominent architectures of a computer networking which has been suggested for purpose of enhancing the QoS. In this architecture, which is illustrated in Figure 2.1 , traffic is divided into different traffic classes. The main approach of the DiffServ is based on a simple model, which classifies IP packets to different classes and defines the allocation of a certain amount of forwarding resources for each class (?). The authors of this paper proposed a new algorithm for pre–marking packets in DiffServ which reflect the relative importance inside an SVC video stream over DiffServ. (?) analyzed and compared the end-to-end QoS performance parameters of FTP and CBR traffics. The parameters which are used in this paper are delay, jitter, loss ratio and throughput, using different types of Diffserv policies (i.e. Token Bucket, TSW3CM and SRTCM). The authors found that performance depends on the kind of Diffserv policy which is used by the traffic.

DiffServ has the potential to gain additional revenue from customers as a premium for guaranteed, or enhanced QoS (?). The traditional DiffServ has a small number of classes plus the BE class as illustrated in Figure 2.2. However, to support these classes there must be a mechanism for charging customers. In the case of VOIP services provided by LTE this mechanism exists. However, charging customers for better QoS for services other than voice is likely to be difficult. Any opportunity

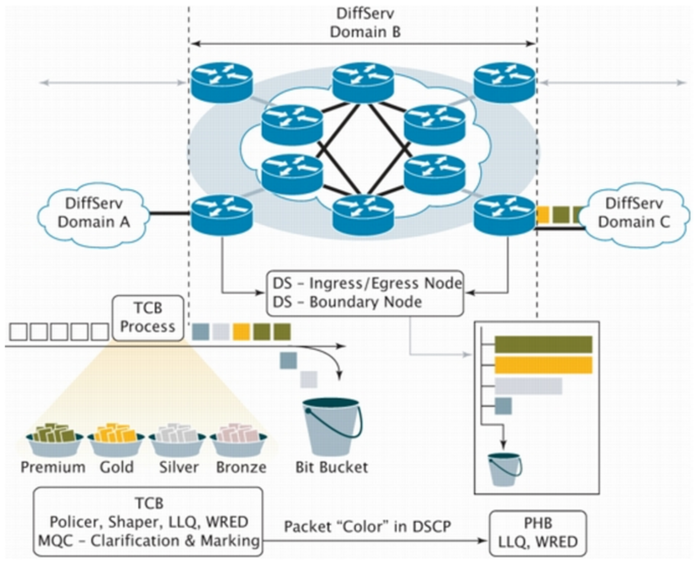


Figure 2.1: DiffServ Architecture from (?)

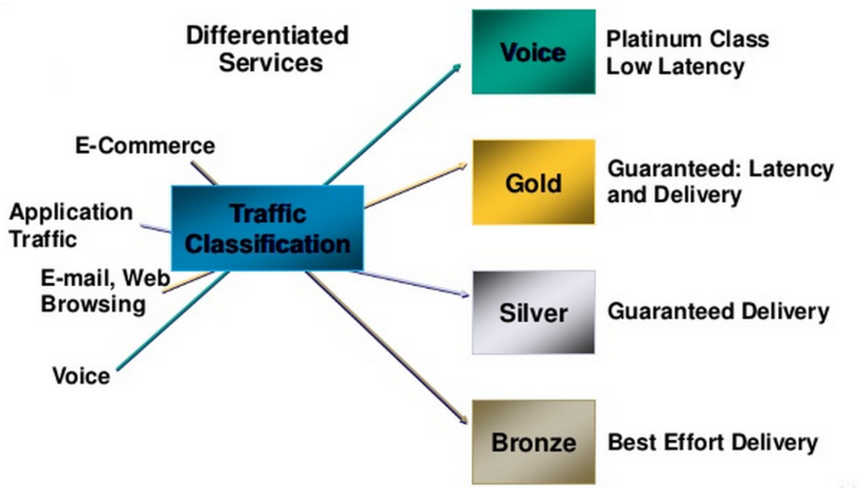


Figure 2.2: DiffServ Classes

from (?)

to provide services with a premium price is attractive to Service providers, but it remains to be seen whether customers will find this concept appealing. In comparing the total cost and benefits of DiffServ, its potential to generate revenue needs to be taken into account. Also, the revenue generation role of DiffServ can potentially undermine its utility as a mechanism for managing QoS.

The issue of dynamic QoS management in a DiffServ network is the focus in (?). The authors of this paper implemented and evaluated a policy-based management system and they claimed that the multimedia applications performed better with the proposed approach. In (?), the authors claimed that the DiffServ architecture with very conservatively provisioned services can be unstable even in the simplest network topologies. Thus, DiffServ cannot satisfy user requests under all circumstances.

The traditional DiffServ will cover only a small proportion of all traffic with high priority classes, Assured and Expedited classes, because DiffServ focuses on importance rather than size of traffic. Therefore it is ignoring a large proportion of traffic in the BE class. The scheme, termed DefServ, introduced in Chapter 5, responds to the traffic dynamically. This means that traffic management will include mechanisms which benefit traffic in the BE class. Therefore this traffic will gain some protection.

In this thesis, in Chapter 5, we describe and test a new approach to better service based on the understanding of social network behaviour gained in this work. This method is complementary to DiffServ. We call this new method Defensive Services (DefServ) because rather than focussing on differentiation based on previously nominated preferences, it focuses on protection of normal traffic, based on observation and detection of unusual traffic. It is uses a new queue discipline called mice and elephants (MAE).

Chapter 3

Social Network Behaviour Inferred from O-D Pair Traffic

In this chapter, the social network behaviour is inferred from O-D pair traffic and the analysis methods which will used here fall into two groups, the first being based on frequency analysis and second method being based on the use of traffic matrices, with the latter analysis method being further sub-divided into groups based on the traffic mean, variance and covariance. The frequency statistics of origin, destination and O-D Pairs exhibits heavy tailed behaviour. Much of the material in this chapter has was treated in (**?**) and (**?**).

3.1 Introduction

Because traffic data provide quantitative measurements of the intensity and quantity of communication between the entities (predominantly users and servers) which share access to a network, we should be able to estimate social network behaviour from traffic samples (trace files). The key difference between social network behaviour, as an interpretation of traffic, vs traditional traffic theory, is that it emphasizes end-to-end patterns, rather than traffic behaviour over time. For example, in traditional traffic theory, it is conventional to view a flow as equivalent, or a generalization, of a TCP flow. Thus, it has a start and an end. But in the analyses undertaken here, the start and end are not considered important. All the traffic between a certain originating IP address and a different destination IP address are viewed as a flow.

Social network behaviour is about who is talking to who, how much they are saying, and the patterns within these conversations. There are many potential patterns, so we need to focus on certain key patterns initially, for example, whether there exist groups of correlated sources, or of destinations, or of O-D flows.

For reasons of privacy, the true identity of the users represented in a traffic trace is not normally available. In any case, the term *social network behaviour* is not used to refer to the identification of specific connections or relationships between users. In this chapter *social network behaviour* is defined to mean *statistical characteristics* which result from the *variation in connection and quantity of communication between the members* of a community. An example of social networking behaviour which might be expected, and which is discovered in the traffic traces in this study, is the heavy-tailed character of the distribution of the total traffic between each origin-destination pair (O-D pair); that is to say: a small number of O-D pairs carry a large quantity of traffic, and a large number of O-D pairs carry relatively little traffic. This heavy-tailed behaviour is not just observed to occur, but its specific “shape” can, and has been, estimated.

We seek to interpret the traffic, even on one link, as a guide to the behaviour of the whole network. If the observed link is in the core network, the range of IP addresses which appear on it will be very large, and although the traffic observed is a sample, because the features being observed are concerned with statistics of aggregate behaviour, rather than statistics concerned with specific users, it was not felt to be necessary to consider the possibility of bias which might be caused by this sampling.

3.2 Source of data

In this chapter,and the next, traffic data from Center for Applied Internet Data Analysis (CAIDA) (?) is analysed by a variety of methods to discover features of the social network from which the traffic is generated. The IP addresses in CAIDA datasets are anonymized. According to (?), the Crypto-PAn tool was used to anonymize the IP addresses in the dataset. Crypto-PAn is a cryptography-based sanitization tool for network trace owners to anonymize the IP addresses in their traces in a prefix-preserving manner (?). This tool has the following properties: (a) the IP address anonymization is prefix-preserving. In other words, if two original IP addresses share a k-bit prefix, their anonymized mappings will also share a k-bit prefix; (b) the same IP address in different traces are anonymized to the same address, even though the traces might be sanitized separately at different time and/or at different locations.

The CAIDA datasets come from two different locations in the United States, were taken in the years 2014, 2015, and 2016, and were sampled at certain specific dates and times. Each set has 10 data files, and each file has about 5 millions packets captured during one minute. The name of each file is formed by concatenating these details, as listed in Tables 3.2 and 3.3 together. The specific combination of files used in the preparing the figures presented later in this chapter, and in Chapter 4, are also identified by using the details of the data, as described in Tables 3.2 and 3.3.

3.3 Traffic Analysis Methods

3.3.1 Frequently used IP addresses

A traditional *traffic matrix*, which was used for many years in telecommunication networks, contains, for each origin-destination pair, the “volume” of traffic (in Erlangs, or bits/sec), between the origin and the destination. By assigning a common column and row index to each node in the network, such data can be readily presented as a matrix. However, when monitoring Internet traffic, without complex processing to translate the IP addresses into node identities relative to a specific network, it may be difficult to identify the origin and destination *nodes* of packets. Using IP addresses in place of origins and destinations seems the logical step to overcome this problem, but the total number of IP addresses, even in IPv4, is too large to allow matrices with one row and column for each IP address.

For this reason, in order to analyse the traffic in a capture file such as those considered in this chapter (**?**), an essential first step is to identify the *frequently used IP addresses*.

Finding the relatively small number of IP addresses which occur a significant number of times in a capture file has been achieved as follows:

The total number of distinct IPv4 addresses is approximately 4 billion, hence even though IP addresses are integers, we can’t use them as the indices of a vector or matrix. Even in quite a short traffic trace file, the total number of distinct IP addresses can easily exceed 1,000,000, and so it is not feasible to construct and manipulate vectors or matrices to describe the statistics of traffic from one IP address to another, even if this is our underlying objective.

So, instead, when matrices indexed by IP addresses are needed, instead of the IP address itself, as the index, we use its *rank*, and instead of including *all* IP addresses as potential indices, we only use the *most important*. IP addresses are all ranked, by their importance, measured as number of bytes sent from or to that address, and this rank is used both to select which IP addresses need to be included in the traffic matrix and the order in which they are included.

3.3.2 Volume statistics

The total number of origins, destinations, or O-D pairs in the pcap file or collection of pcap files being analysed may be more than we are able to store information about. However, it will be satisfactory if the only sources, destinations, or O-D pairs about which information is not collected are ones for which the total bytes, associated with this origin, destination, or pair, is small. By storing pointers to origins, destinations, or O-D pairs in a heap which is ordered so that the entity with the least total bytes, over an interval of time exceeding a certain threshold, can be easily removed, it is possible to identify origins, destinations, or O-D pairs which can be removed from the list of those about whom statistics are being collected without having to collect the information about this object, except for short periods of time. The volume information collected in this way is stored in such a way that we can estimate the distribution of bytes associated with each origin, destination, or O-D pair.

3.4 Antraff software

In order to carry out the analysis methods described above, and other methods, a software package called *Antraff* (**?**) has been developed. The Antraff package, primarily written in C++, comprises 17 header files, 34 c files, and 12 classes. There

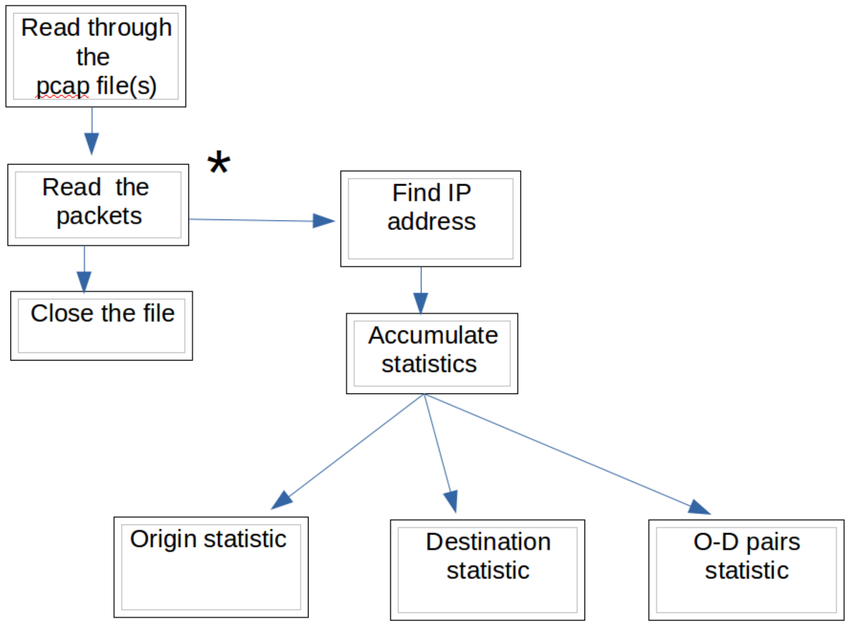


Figure 3.1: Algorithm for collecting origin/destination/O-D pair byte frequencies

are 9,500 lines of code, mainly in C++, but including awk and bash scripts, and a makefile which includes as targets the types of graphs which are frequently needed in analysing traffic.

The central class in this package is a template class called `ipbin`, this template is provided in Appendix A. The fact that a template class with parametrized address object is used ensures that the Antraff package can be used in a flexible way, as the type of analysis which needs to be undertaken evolves. Figure 3.1 shows how this software collects statistics of the following:

(i) Byte frequencies of origins, destinations, or O-D pairs.

(ii) Accumulation of matrices of end-to-end traffic, followed by analysis of these matrices, by singular-value decomposition. There are three different ways in which traffic matrices can be accumulated, which can be intuitively described as mean, variance, and covariance matrices.

3.5 Experiments

In this section we present five experiments. In the first experiment, in Subsection 3.5.1, we find that the distribution of bytes per source/destination is heavy–tailed. Also in this subsection, we calculate the Pareto shape parameter of these curves. In Subsection 3.5.2, the frequency curves of source, destination and O–D pair are presented together and compared. Subsection 3.5.3 estimates the social network traffic behaviour in another way and the diversity or concentration of the community of users is measured. In Subsection 3.5.5, a log-normal property of the distribution of O–D flow sizes is observed. The process of conducting these experiments, and the experiments of Chapter 4, is explained in Appendix B by an example.

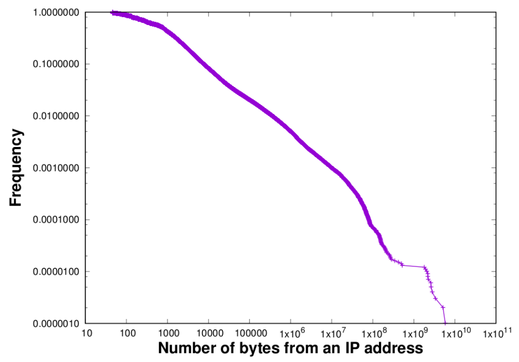
3.5.1 Source and Destination Byte Frequencies

The traffic associated with individual IP addresses exhibits the Pareto principle, which is that a small proportion of IP addresses is associated with a high proportion of the traffic. In other words, the distribution of bytes per destination/source is *heavy–tailed*. This is illustrated in Figures 3.2–3.4.

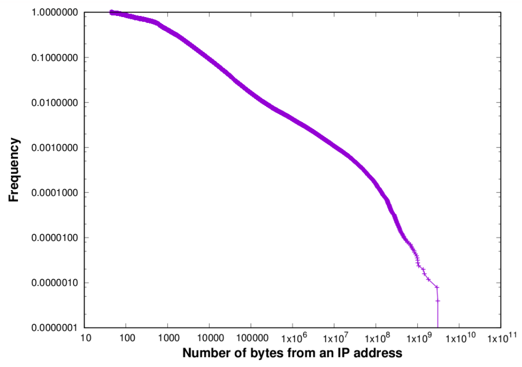
On a log-log scale, these plots are approximately linear. All curves exhibit a tendency to "roll off", i.e. the curve falls away from the linear shape to the right. If the population was such that this plot was perfectly linear on a log-log scale, in a theoretical sense for the whole population, the distribution of the quantity considered would be Pareto.

Figures 3.2–3.4 *suggest* that most bytes are accounted for by the larger sources of traffic. However we can't actually read from these plots the proportion of bytes accounted by any initial portion (in decreasing weight) of sources. Figure 3.6 and

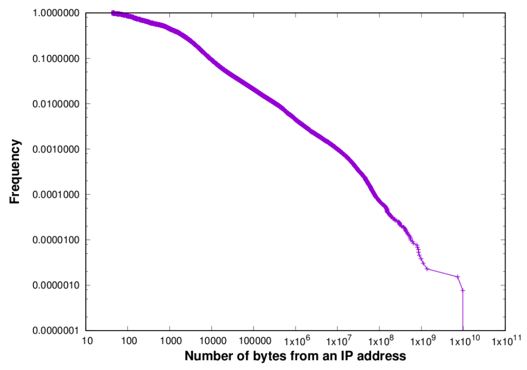
3.7 are designed to allow this. These figures present the same information, but now the x-axis is the cumulative proportion of source and destination IP addresses respectively, in the data set, in order of weight, and the y-axis is the cumulative proportion of total bytes accounted for, by these IP addresses.



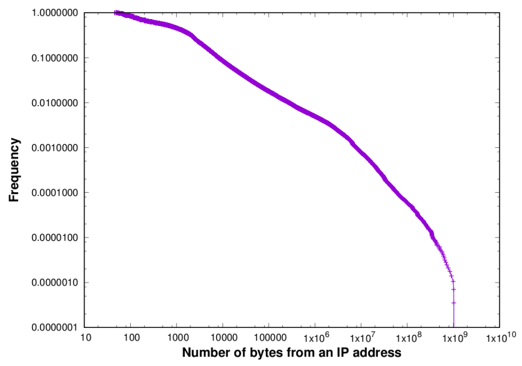
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

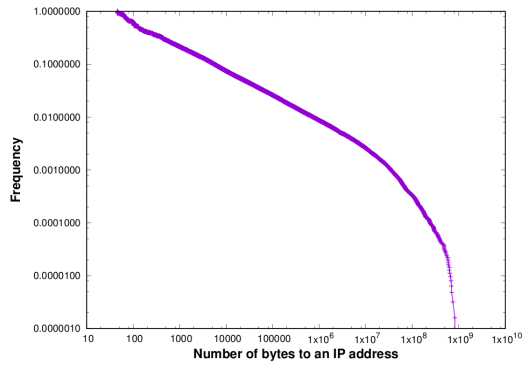


(c) Trace used: Chicago 2015 data-set

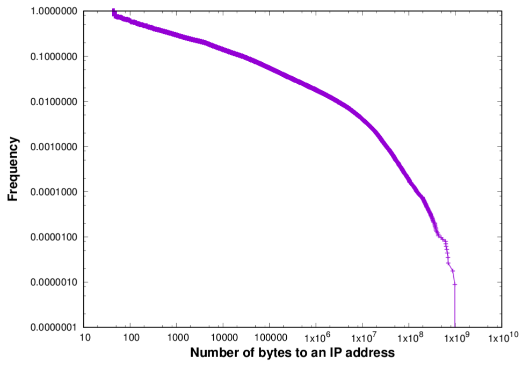


(d) Trace used: Chicago 2016 data-set

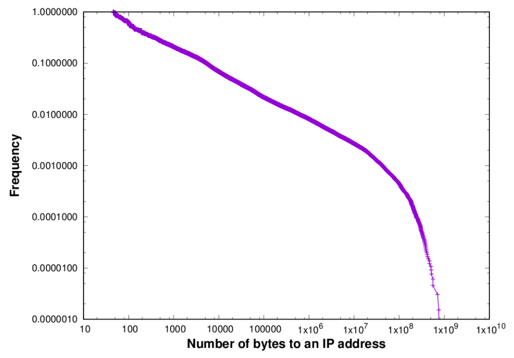
Figure 3.2: Bytes per source for different data–sets.



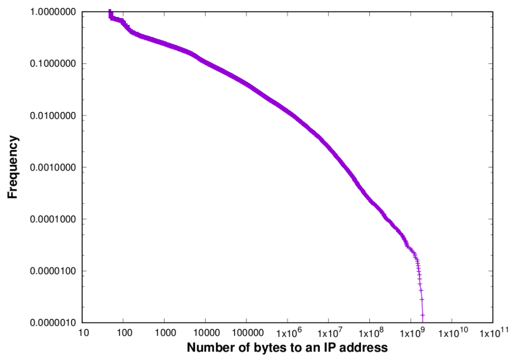
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

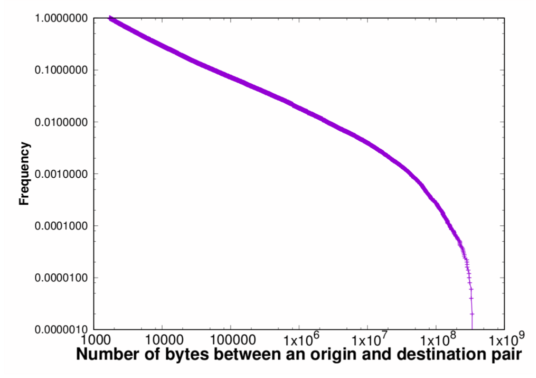


(c) Trace used: Chicago 2015 data-set

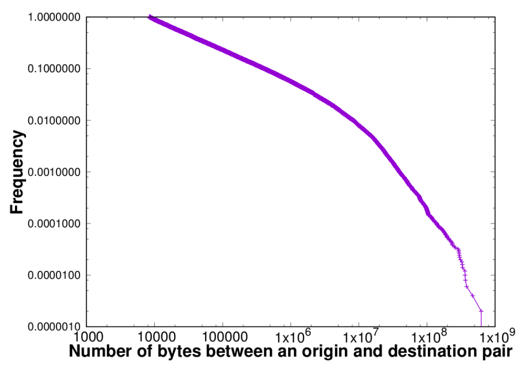


(d) Trace used: Chicago 2016 data-set

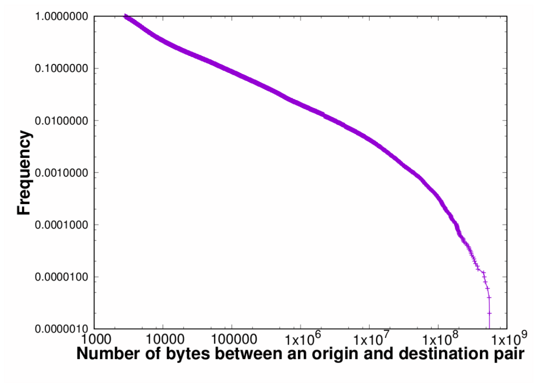
Figure 3.3: Bytes per destination for different data–sets.



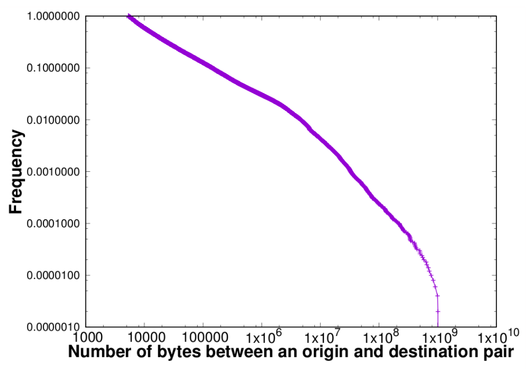
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set



(c) Trace used: Chicago 2015 data-set



(d) Trace used: Chicago 2016 data-set

Figure 3.4: Bytes per O–D pairs for different data–sets.

These graphs confirm the intuitive idea already introduced, that a small proportion of source IP addresses accounts for a large proportion of traffic, in bytes. However, the story is not *extreme.* To account for 90% of the bytes it is necessary to include at least 1% of sources, destinations, or O-D pairs, which is still quite a large number – approximately 2,000 in for the traces used in Figures 3.6, 3.7 and 3.8.

Estimating γ

Another way to show that the traffic exhibits the Pareto principle is by calculating γ values (the shape parameter of the Pareto distribution, sometimes also referred to as α), i.e. the slope, on a log-log plot, of the frequency vs the number of bytes for each methods; source, destination and O–D pairs. Table 3.1 contains the values of γ for all the curves which are shown in Figures 3.2–3.4. All of these values are between 1.5 and 2. The values for the destinations is most frequently higher than those for sources and O-D pairs. The values for the four datasets are fairly consistent, although the Chicago 2016 dataset has somewhat smaller values than the others.. The value of γ is a measure of how *heavy* are the tails of the Pareto distribution in this case. The approach of estimating γ is a statistically robust method because it does not rely on identifying flows in the traffic (e.g. by identifying them with tcp flows).

For prediction of future events, only a certain range of values in these graphs is important and should therefore be considered when the slope is calculated. Another likely reason for these curves to "roll off" in the region of large flow sizes is that since networks are not able to provide satisfactory performance for such large flows, users will be less likely to request them. Figure 3.5 shows how the graph is divided into three different regions; R_0, R_1 and R_2. The most important region is R_1. R_0 is unimportant because the flow sizes of this region are very small and R_2 is unreliable due to the smallness of sample.

chapter3graphs/slope.png

Figure 3.5: A sample distribution

To predict how frequently the large flows (*elephants*) occur, observe that, if flow sizes larger than some value follow a Pareto distribution,

$$P\{F > f \mid F > \delta\} = (f/\delta)^{-\gamma}, \tag{3.1}$$

where: F is the random flow size, f is threshold value for which the probability is being evaluated, and δ is an arbitrary flow size value which is smaller than any which are currently of interest.

For example, in Figure 3.4(b), let us take $\delta = 10^7$. This trace is of length 10 minutes, and the number of flows included in the analysis leading to the figure is 500,000, so the total rate of arrival of flows (including on the ones analysed), is 50,000 per minute. According to the graph, the frequency of flows larger than 10^7 is 0.01, so the rate of arrival of flows longer than 10^7 is 500. Now let us suppose we want to know how often flows of size 10^8 will arrive (i.e., we regard these as the elephants). By (3.1), the frequency of these flows will be $\left(\frac{10^8}{10^7}\right)^{-\gamma} \times 500 = 9.3$ flows per minute (using $\gamma = 1.73$, as given in Table 3.1). Flows of size 10^9 or larger will occur, using the same method, $\left(\frac{10^9}{10^7}\right)^{-\gamma} \times 500 = 0.173$ per minute, or, putting it more naturally, once every 7 minutes.

Another reason, besides the smallness of the sample, for the "roll-off" at the right-hand end of the graphs in Figure 3.4 is that although users might make larger and larger requests, the network through which these requests must pass is unable to carry them. This factor means that the prediction of the frequency of large flow sizes provided by (3.1) will be less accurate for larger flow sizes. However, if the roll-off is due to network limitations, the prediction is not inaccurate as a prediction of user behaviour, but rather as a prediction of network performance.

Table 3.1: The values of γ for curves of frequency vs. number of bytes for each method.

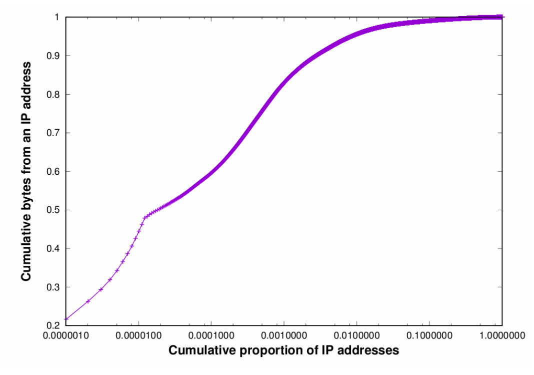
Data-set	Type of analysis	γ value (Pareto shape parameter)
CAIDA-Chicago-2014	Source	1.65
CAIDA-Chicago-2014	Destination	1.73
CAIDA-Chicago-2014	O–D pair	1.66
CAIDA-San Jose-2014	Source	1.78
CAIDA-San Jose-2014	Destination	1.91
CAIDA-San Jose-2014	O–D pair	1.73
CAIDA-Chicago-2015	Source	1.63
CAIDA-Chicago-2015	Destination	2.001
CAIDA-Chicago-2015	O–D pair	2.005
CAIDA-Chicago-2016	Source	1.7
CAIDA-Chicago-2016	Destination	1.51
CAIDA-Chicago-2016	O–D pair	1.63

Table 3.2: CAIDA data-sets (each data set consists of 10 data files, as in Date/Time column, each file has same prefix and same suffix- part 1)

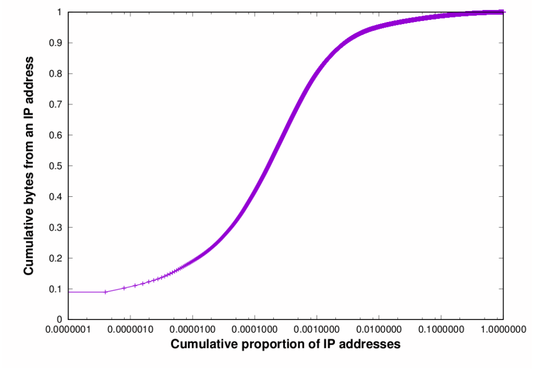
Data-set Name	**Prefix**	**Date/Time**	**Suffix**
Chicago 2014	equinix-chicago-.dirA-2014	0320-130000, 0320-130100, 0320-130200, 0320-130300, 0320-130400, 0619-130000, 0619-130100, 0619-130200, 0619-130300, 0619-130400	UTC-.anon-.pcap
San Jose 2014	equinix-Sanjose-.dirA-2014	0320-125905, 0320-130100, 0320-130200, 0320-130300, 0320-130400, 0619-130000, 0619-130100, 0619-130200, 0619-130300, 0619-130400	UTC-.anon-.pcap

Table 3.3: CAIDA data-sets (each data set consists of 10 data files, as in Date/Time column, each file has same prefix and same suffix- part2)

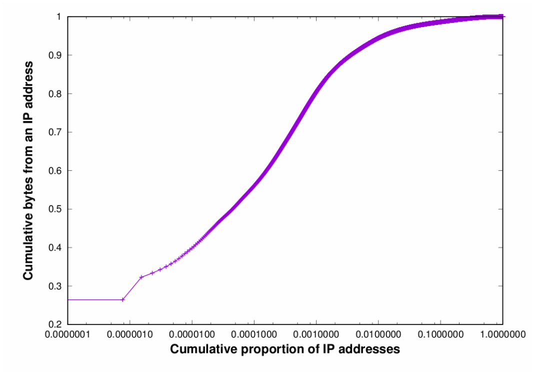
Data-set Name	Prefix	Date/Time	Suffix
Chicago 2015	equinix-chicago-.dirA-2015	0219-125911, 0219-130000, 0219-130100, 0219-130200, 0219-130400, 0917-125911, 0917-130000, 0917-130100, 0917-130200, 0917-130300	UTC-.anon-.pcap
Chicago 2016	equinix-chicago-.dirA-2016	0121-125911, 0121-130000, 0121-130100, 0121-130200, 0121-130300, 0121-125911, 0317-130000, 0317-130100, 0317-130200, 0317-130300	UTC-.anon-.pcap



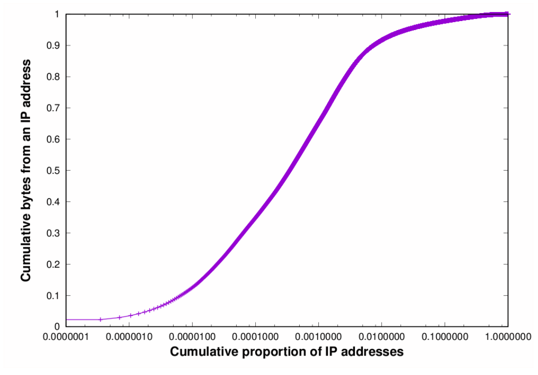
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

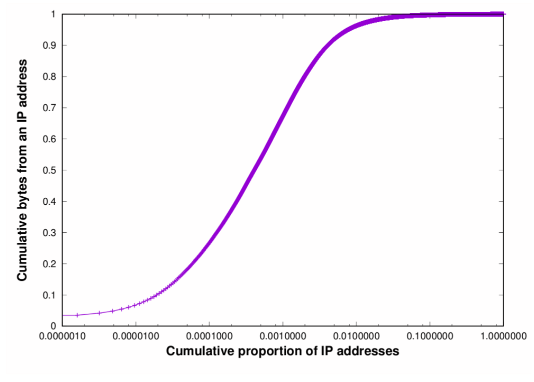


(c) Trace used: Chicago 2015 data-set

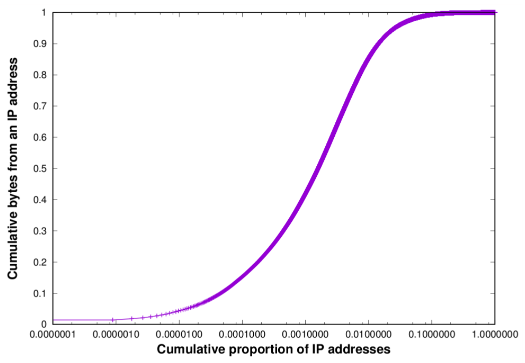


(d) Trace used: Chicago 2016 data-set

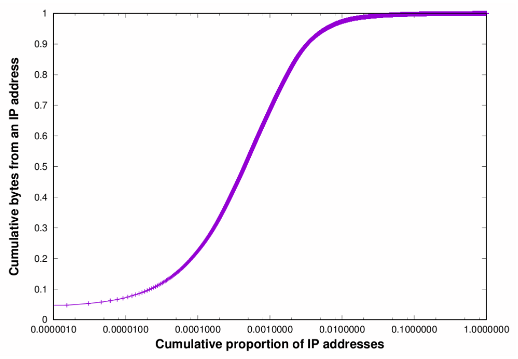
Figure 3.6: Cumulative sources bytes from an IP address vs Cumulative proportion of IP addresses for different data–sets.



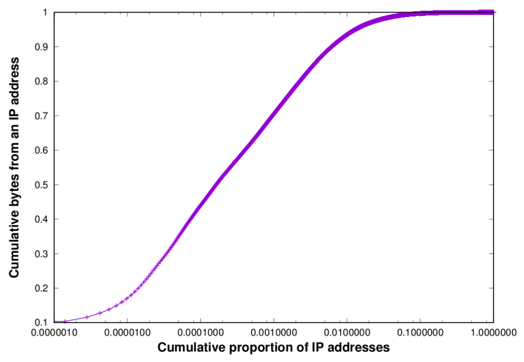
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

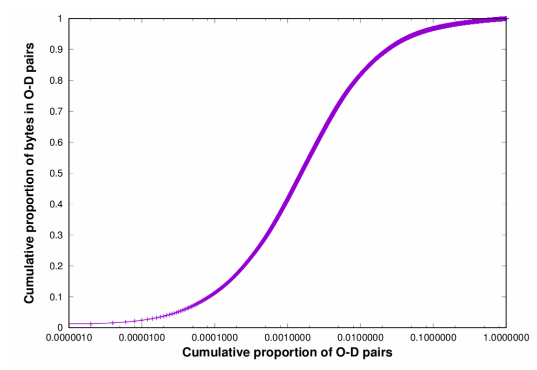


(c) Trace used: Chicago 2015 data-set

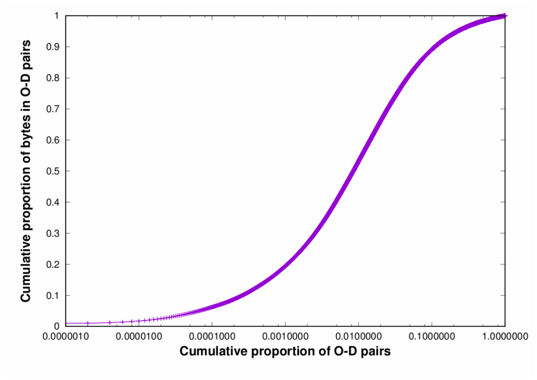


(d) Trace used: Chicago 2016 data-set

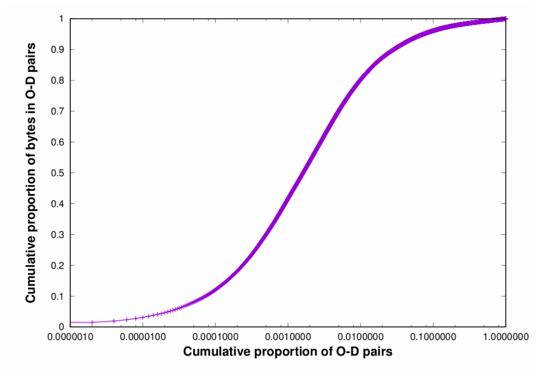
Figure 3.7: Cumulative destination bytes to an IP address vs Cumulative proportion of IP addresses for different data–sets.



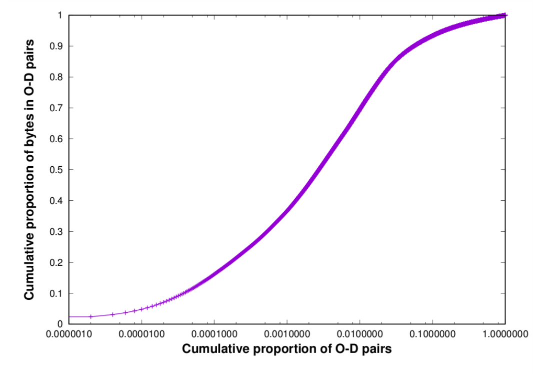
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set



(c) Trace used: Chicago 2015 data-set



(d) Trace used: Chicago 2016 data-set

Figure 3.8: Cumulative bytes in O-D pairs address vs Cumulative proportion of O-D pairs for different data–sets.

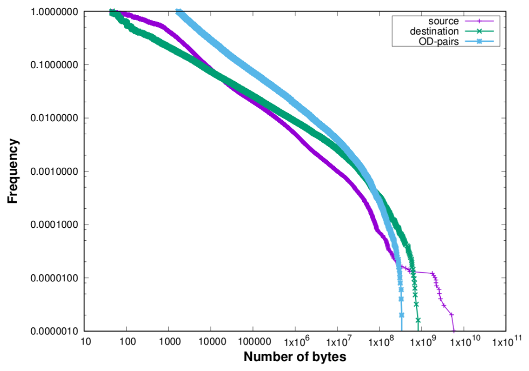
Another way to analyse the traffic is to find the relative proportions of bytes associated with different O-D pairs. A graphs which illustrate this approach are shown in Figures 3.6–3.8. This time, instead of focussing on which *sources* or which *destinations* account for the traffic, it is the *O-D pairs*. We are looking here for an understanding of the structure of traffic from the point of view of O-D pairs which may prove to be very important in the development of an end-to-end traffic model.

The statistical results of O-D pairs, in this chapter, are different from other studies in regard of the number of IP addresses which are analysed. The authors of (?) have analysed the structure of Origin-Destination traffic in a trace of Internet backbone traffic by using PCA to systematically decompose the structure of OD flow time series. They describe the trace they analyse as including hundreds of origins and destinations, and hundreds of flows. Another group of authors in (?) have analysed 1000 IP addresses to study the traffic characteristics of O-D pairs components, whereas the ones considered in this chapter includes hundreds of thousands of origins and destinations, and millions of flows. Another difference is that the traces considered in this chapter have durations of approximately one minute, whereas in (?) they consider traces lasting hours, days, or weeks. Figures 3.6–3.8 illustrate the relation between cumulative bytes in O-D pairs vs cumulative proportion of O-D pairs starting with the largest flows because of the importance of these flows. These Figures show that the first 1% of O-D pairs covers about 90% of bytes.

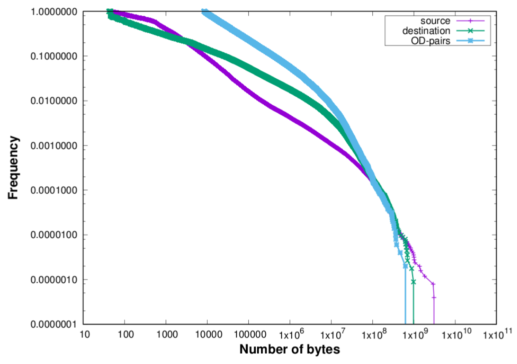
3.5.2 Comparison of source, destination and O–D pairs

Figures 3.9 and 3.10 show the frequency curves of source, destination and O–D pairs on the same graph. These graphs show the difference between the frequency curve of source, destination and O–D pair in CAIDA data and simulation experiment data. The destination curves, prior to where roll-off, appear to exhibit lower slope than the source curves, i.e. they look flatter, in other words, γ is smaller. On the other hand, the slope of O–D pair curves and the slope of source curves appears to be very similar Figure 3.9. This provides further confirmation for the existence of non-trivial eigensources and eigendestinations because the simulated traffic, which is formed by independent Poisson streams of Pareto distributed flows, does not exhibit (in Figure 3.10) a systematic difference like this.

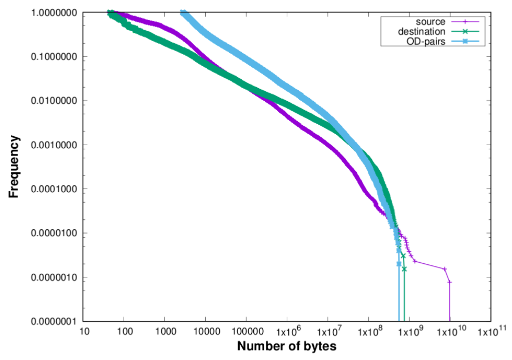
Figure 3.10 shows that the frequency curves of the source, destination and O–D pair are the same. The simulated model has Pareto traffic from whole collection of origins, destinations and O–D pairs. This model does not have eigensources and eigenditenations, it has merely independent Pareto distributed flows between all origins and destinations.



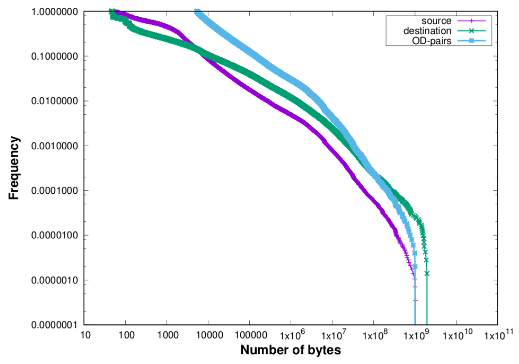
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set



(c) Trace used: Chicago 2015 data-set



(d) Trace used: Chicago 2016 data-set

Figure 3.9: Bytes per source, destination and O–D pair for different data–sets.

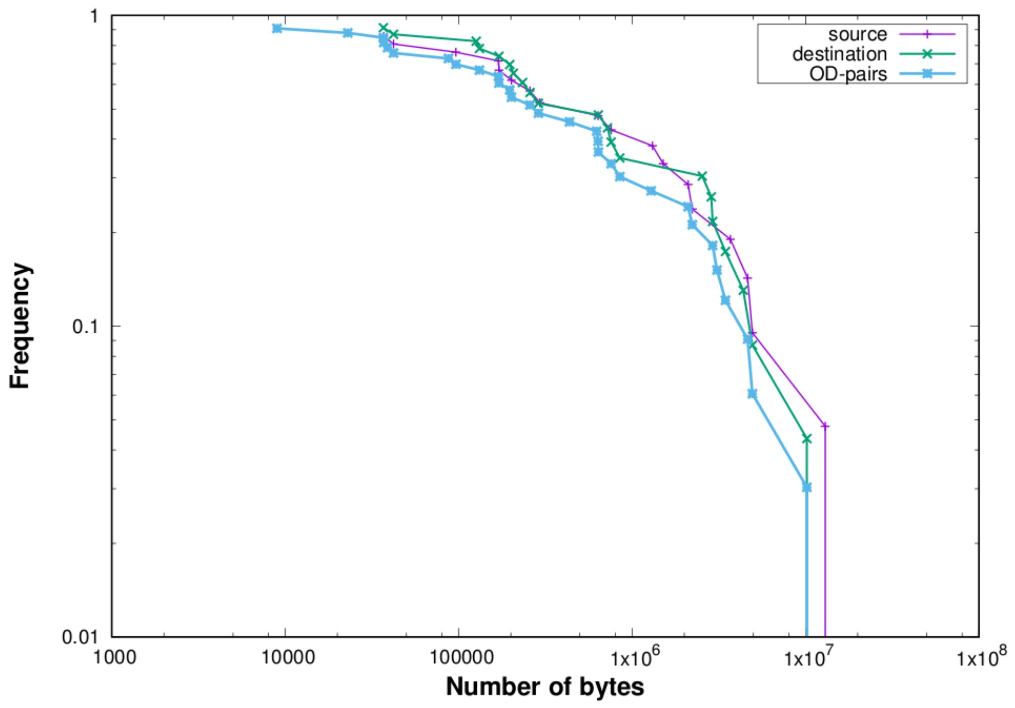


Figure 3.10: Bytes per source, destination and O–D pair for simulation data

3.5.3 O-D-pairs vs origins and destinations

Figure 3.11 exhibits another way to estimate social network traffic behaviour from a traffic trace. This plot was generated by scanning through a traffic trace, and at each point of the trace generating a point in the plot by using the total number of sources (destinations) found so far in the trace as the x-coordinate, and the total number of O-D flows found so far as the y-coordinate. The traffic traces are captured from Chicago and San Jose on 2014, 2015 and 2016 in US. This has been plotted using log scales for both axes. The curve is approximately linear, with a slope close to 1. The slope of this curve, on a log-log scale, is a measure of the diversity or concentration of the community of users (and servers) represented in the trace. The larger the slope the broader (more diverse) is the community whose communication is taking place in the trace file.

The closeness to 1 indicates the degree to which communication in this trace is within closed communities. In other words, the closeness of the slope in these Figures to 1 means that the individuals represented in the trace tend to communicate with the same relatively small group of friends.

3.5.4 Experimental results of the plots slope

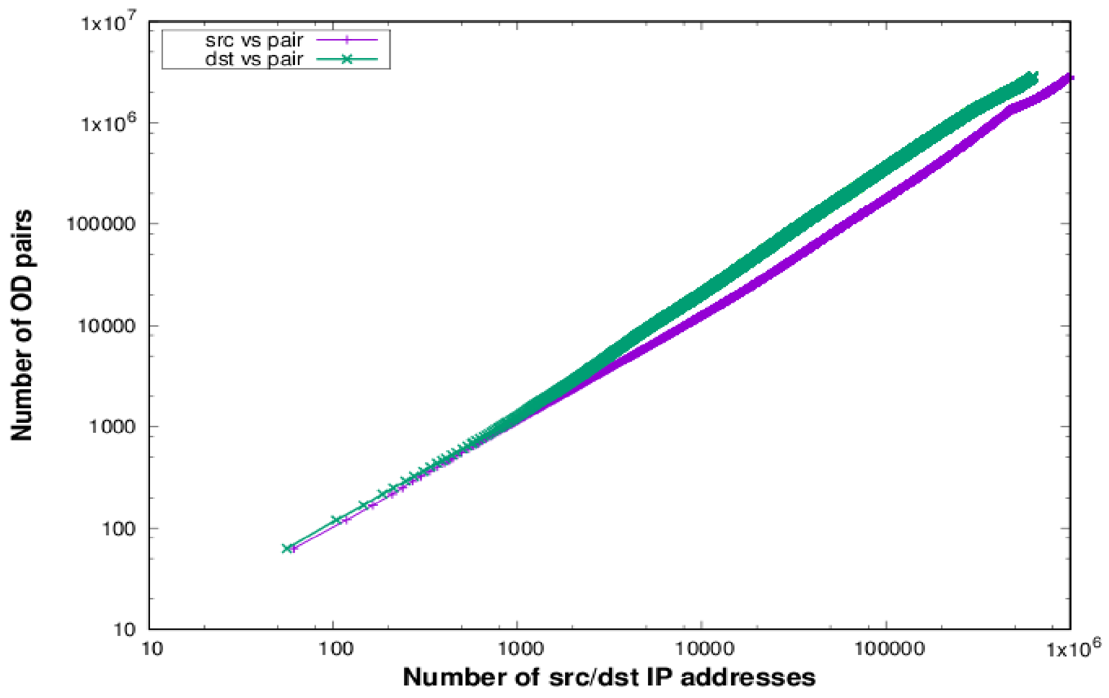
The slopes in Figure 3.11 which are represent the relation between the number of sources and the number of destinations in x-axes and the number of O-D pairs in y-axes, are shown in Table 3.6.

If everyone communicated with only one other individual, the slope would be exactly 1, while if everyone communicated with everyone else, the slope would be 2. The slope will also be very close to 1 if there is a small number of central nodes, which we might call the gatekeepers, and every other individual communicates only with these gatekeepers. The data files, which are downloaded from CAIDA, in Table 3.6 come from different years, 2014, 2015 and 2016, and different places, Chicago and San Jose in the US.

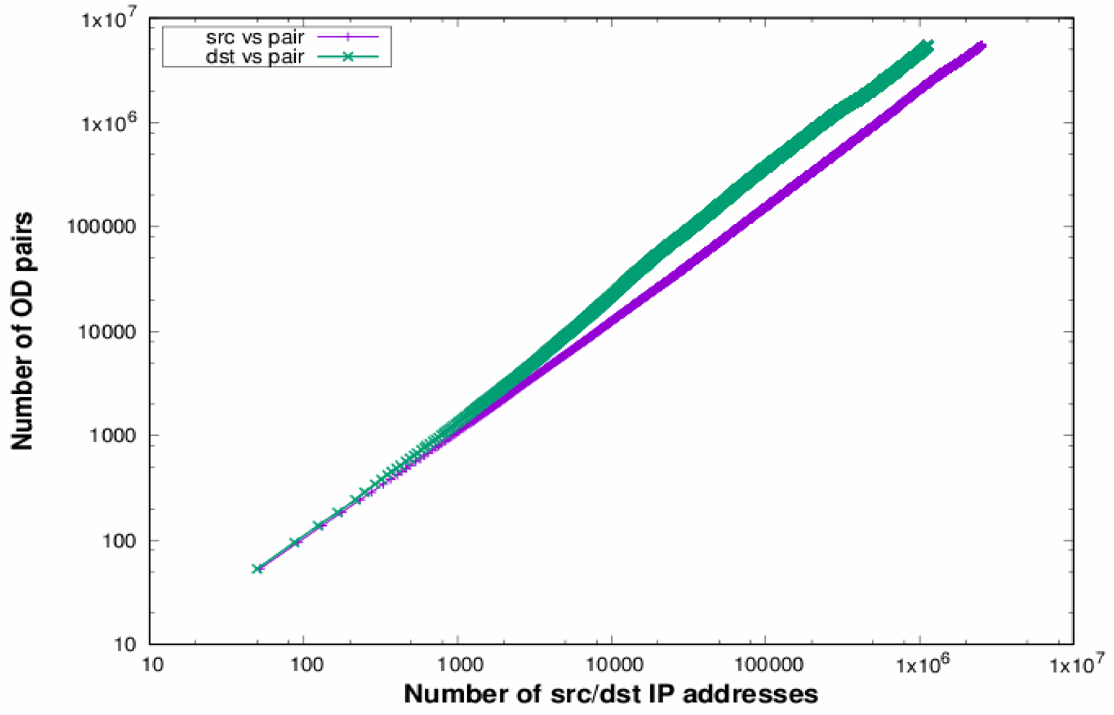
Given the prominence of key sites in the Internet, the likes of Google, Youtube, and Facebook, it is therefore not surprising that the slopes in these Figures are close to 1 as shown in Table 3.6.

To investigate further the significance of the slopes, a t-test was carried out to compare one group against another and the results of these tests are as follows.

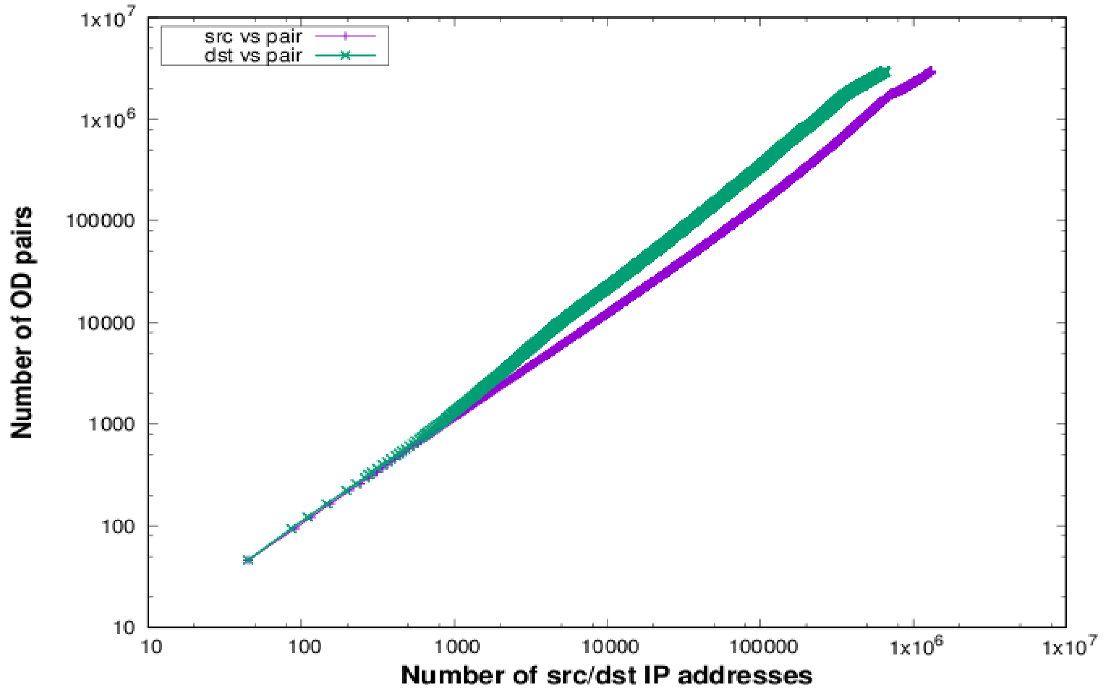
(i) the slopes of the source plots are less than those of the destination plots;



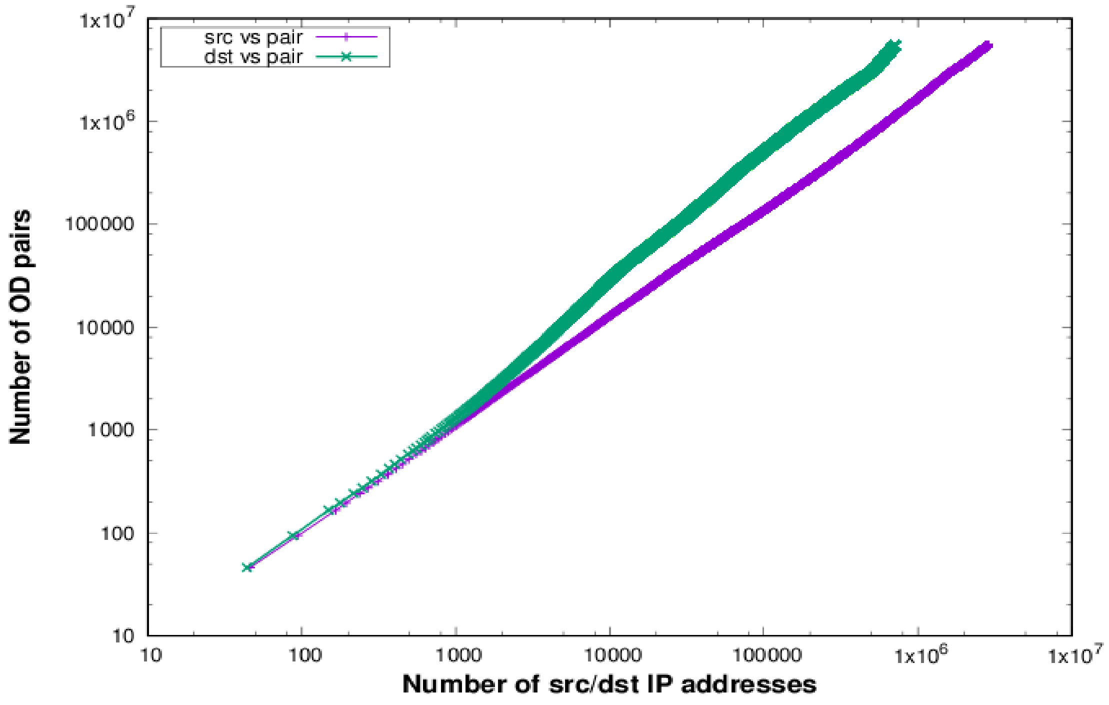
(a) [Trace used: Chicago 2014 data-set]



(b) [Trace used: San Jose 2014 data-set]



(c) [Trace used: Chicago 2015 data-set]



(d) [Trace used: Chicago 2016 data-set]

Figure 3.11: Number of O-D pairs vs. Number of source and destination IP addresses for different data files[see Table 3.2]

(ii) the four source slopes are different from each other;

(iii) the destination slopes in 2014-2015 are not significantly different, but the destination slope in 2016 is significantly larger than those in earlier years.

All the t-tests were un-paired and tested equality vs inequality except for the comparisons of source slopes vs destination slopes and the comparison of the Chicago 2016 destination slope vs the other destination slopes. The test of destination slopes vs source slopes was paired and tested the hypothesis that the source slope is less than the destination slope vs the alternative that it is not less. The test of the Chicago destination slope vs the other destination slopes was unpaired and tested the hypothesis that the Chicago 2016 slope is greater than the other source slopes.

3.5.5 O-D Flow sizes

It has been suggested previously that flow sizes in the traditional sense of the number of bytes in an individual TCP connection have a power-law distribution (?). Many studies have been written analysing traffic models based on this assumption, e.g. (?). However, in this chapter the term *flow* has a different interpretation: it refers to all the traffic between a certain originating user and another destination user. The power-law character of the sizes of *this* type of flow does not necessarily follow from the power-law character of flows of the type considered in (?). Nevertheless, the broadly linear shape of the curves in Figures 3.6–3.8 supports the power-law character of flows in the sense this term is used here.

Figure 3.12 is the same as Figures 3.6–3.8 except that the vertical scale has been transformed in the same manner as a quantile plot, of the type used to check normality of a distribution. The straightness of the plot in Figure 3.12 indicates that the rank of an O-D pair selected by drawing a byte randomly, from all the bytes

Table 3.4: Source and destination slope values of traffic traces – part 1

Data-set Name	**File Name**	**Slope–S**	**Slope–D**
Chicago 2014	equinix-chicago.dirA.20140320-130000.UTC.anon.pcap	1.19825	1.240841
	equinix-chicago.dirA.20140320-130100.UTC.anon.pcap	1.206898	1.240961
	equinix-chicago.dirA.20140320-130200.UTC.anon.pcap	1.196147	1.231688
	equinix-chicago.dirA.20140320-130300.UTC.anon.pcap	1.212603	1.23856
	equinix-chicago.dirA.20140320-130400.UTC.anon.pcap	1.205314	1.245019
	equinix-chicago.dirA.20140619-130000.UTC.anon.pcap	1.151398	1.156585
	equinix-chicago.dirA.20140619-130100.UTC.anon.pcap	1.159662	1.152868
	equinix-chicago.dirA.20140619-130200.UTC.anon.pcap	1.146592	1.194499
	equinix-chicago.dirA.20140619-130300.UTC.anon.pcap	1.198673	1.178119
	equinix-chicago.dirA.20140619-130400.UTC.anon.pcap	1.169297	1.236181
San Jose 2014	equinix-sanjose.dirA.20140320-125905.UTC.anon.pcap	1.11139	1.178774
	equinix-sanjose.dirA.20140320-130100.UTC.anon.pcap	1.117446	1.154508
	equinix-sanjose.dirA.20140320-130200.UTC.anon.pcap	1.111833	1.154293
	equinix-sanjose.dirA.20140320-130300.UTC.anon.pcap	1.108536	1.171695
	equinix-sanjose.dirA.20140320-130400.UTC.anon.pcap	1.119749	1.16328

Table 3.5: Source and destination slope values of traffic traces – part 2

Data-set Name	**File Name**	**Slope–S**	**Slope–D**
San Jose 2014	equinix-sanjose.dirA.20140619-130000.UTC.anon.pcap	1.119749	1.216095
	equinix-sanjose.dirA.20140619-130100.UTC.anon.pcap	1.091031	1.211002
	equinix-sanjose.dirA.20140619-130200.UTC.anon.pcap	1.088507	1.207736
	equinix-sanjose.dirA.20140619-130300.UTC.anon.pcap	1.100227	1.221282
	equinix-sanjose.dirA.20140619-130400.UTC.anon.pcap	1.094473	1.222404
Chicago 2015	equinix-chicago.dirA.20150219-125911.UTC.anon.pcap	1.156123	1.22055
	equinix-chicago.dirA.20150219-130000.UTC.anon.pcap	1.153414	1.242492
	equinix-chicago.dirA.20150219-130100.UTC.anon.pcap	1.12059	1.31508
	equinix-chicago.dirA.20150219-130200.UTC.anon.pcap	1.161377	1.234627
	equinix-chicago.dirA.20150219-130400.UTC.anon.pcap	1.160145	1.240479
	equinix-chicago.dirA.20150917-125911.UTC.anon.pcap	1.113016	1.079916
	equinix-chicago.dirA.20150917-130000.UTC.anon.pcap	1.113626	1.06874
	equinix-chicago.dirA.20150917-130100.UTC.anon.pcap	1.114851	1.067286
	equinix-chicago.dirA.20150917-130200.UTC.anon.pcap	1.119336	1.067437
	equinix-chicago.dirA.20150917-130300.UTC.anon.pcap	1.117681	1.085924

Table 3.6: Source and destination slope values of traffic traces – part 3

Data-set Name	**File Name**	**Slope–S**	**Slope–D**
Chicago 2016	equinix-chicago.dirA.20160121-125911.UTC.anon.pcap	1.047565	1.242566
	equinix-chicago.dirA.20160121-130000.UTC.anon.pcap	1.064275	1.236514
	equinix-chicago.dirA.20160121-130100.UTC.anon.pcap	1.062016	1.234313
	equinix-chicago.dirA.20160121-130200.UTC.anon.pcap	1.063316	1.22516
	equinix-chicago.dirA.20160121-130300.UTC.anon.pcap	1.061707	1.237587
	equinix-chicago.dirA.20160317-125911.UTC.anon.pcap	1.088713	1.479135
	equinix-chicago.dirA.20160317-130000.UTC.anon.pcap	1.098664	1.474604
	equinix-chicago.dirA.20160317-130100.UTC.anon.pcap	1.099633	1.458606
	equinix-chicago.dirA.20160317-130200.UTC.anon.pcap	1.095182	1.482734
	equinix-chicago.dirA.20160317-130300.UTC.anon.pcap	1.095915	1.48362

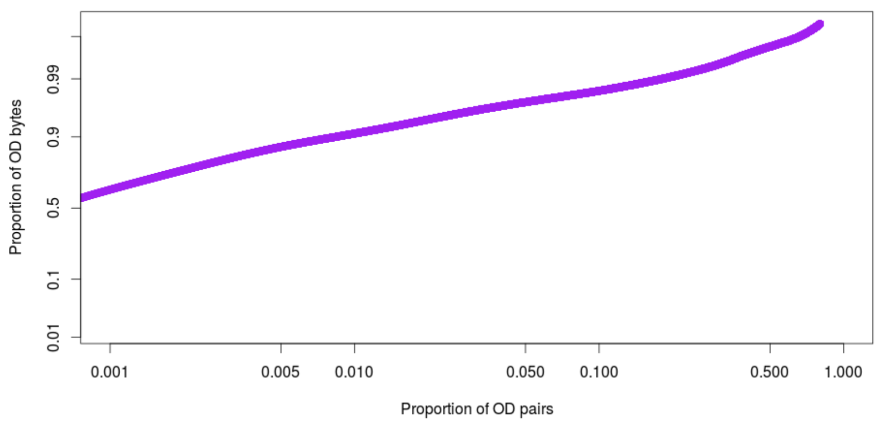


Figure 3.12: Log-normal character of OD flow sizes [Chicago 2015 data-set]

passing through a link, and then finding the O-D pair rank between which it was being transmitted, is approximately log-normally distributed. Note: the smallest 15% of flows are not included in this plot.

3.6 Summary of the chapter

In this chapter, the behaviour of social networks from O–D pair traffic has been inferred by using various methods of traffic analysis. We found that a Pareto principle applies to every aspect of Internet traffic. Most users contribute very few bytes, and most bytes are from a very small proportion of users. The slope of the frequency vs the number of bytes from an IP address, the number of bytes to an IP address and the number of bytes between an origin and destination pair curves on a log-log scale, γ, has been calculated to show how heavy are the tails of the Pareto distribution in each case.

Chapter 4

Spectral Analysis of Traffic Matrices

In this chapter we extend the knowledge and get better understanding of the social networks traffic behaviour by analysing traffic matrices obtained from the CAIDA traces using different spectral decomposition techniques: (a) SVD of the traffic mean matrix; (b) SVD of the traffic variance matrix; and (c) SVD of the OD-pair covariance matrix.

4.1 Introduction

In the mathematics of matrices and linear operators, analysis of matrices or operators in terms of their eigenvalues and eigenvectors is sometimes referred to as *spectral analysis* (**?**). We shall use this term to refer to all methods which rely on identifying the eigenvalues and eigenvectors of matrix representations of experimental data. The Fourier transform, in its various forms, also falls under this heading since it can be viewed as finding the eigenvalues of a linear system associated with the data to which the transform is applied. The insight from spectral analysis is used in physics, computer science, engineering, psychology, education, and virtually every field of science. It is applied by the network research community to analyse Internet traffic for various purposes. According to (**?**), the spectral analysis technique has been used in (**?**), for example, to extract periodic patterns embedded in a trace. Also, the authors of (**?**) and (**?**) used the technique to classify attacks, in (**?**) a signal analysis of network traffic anomalies.

Principal component analysis, in particular, uses spectral analysis of covariance matrices to identify *features* by means of which experimental data can be best explained. This is often used to validate *instruments*, i.e. psychological tests. Pattern recognition algorithms use spectral analysis to decompose observations into features which are then used in further steps of analysis.

Social network traffic is traffic which resulting from all network connections and it is about who is talking to who, how much they are saying, and the patterns within these conversations (**??**). In this chapter we identify the methods which are capable of identifying the key features of a social network by analysing matrices associated with traffic traces. Eigenflows are linear combinations of flows between multiple origin-destination pairs. Figure 4.1 illustrates the phenomenon of eigenflows. The singular value decomposition algorithm identifies the linear combinations which explain the complete data set with optimal least squared error. In Figure 4.1, each stream colour refers to certain eigenflow. In particular, it is not the case that any collection of flows that are lumped together can be called an eigenflow. They must have a statistical relationship, and it is virtually certain that this statistical relationship must be due to a relationship of the underlying network activity.

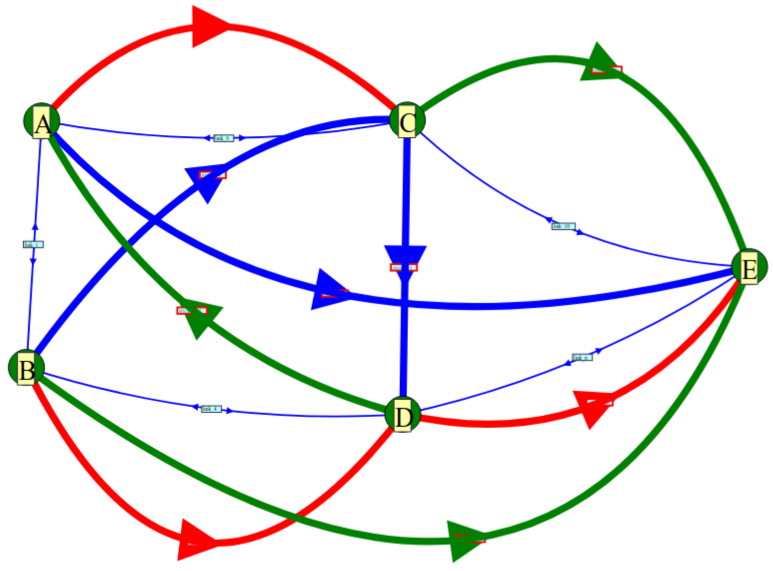


Figure 4.1: The eigenflows in the social networks traffic

4.2 Related work

Eigenflows were first introduced in (?). They analyzed the structure of Origin–Destination traffic traces from two different networks: the European Sprint backbone network and the Abilene Internet2 backbone traffic, by using PCA. They found that set of OD flows has small intrinsic components.these eigenflows reveal important features of network traffic. These features vary in a predictable manner as a function of the amount of traffic carried in the O–D flow.

The eigenflows of Internet traffic (with a significantly different definition of the origins and destinations) were also estimated and discussed in (?) by using singular value decomposition of traffic matrices of several different types. It was not confirmed in either of these papers whether these eigenflows are statistically significant. Such a test is conducted in Section 4.6.2. The main differences between (?) and (?) are: (a) the traces considered in (?) have time intervals of about one minute, while in (?)

the authors consider traces lasting hours, days, or weeks; (b) the matrices which are considered in (?), includes hundreds of thousands of origins and destinations, and millions of flows, whereas those in (?) refer to a much smaller number of origins and destinations; (c) the IP addresses in CAIDA datasets, which is used in (?), are anonymized by Crypto-PAn tool. However, the authors of (?) identify the ingress and egress points of each flow. The methods for determining whether these eigenflows are significant or not, and measuring their significance, is investigated and discussed in this chapter. Estimating the statistical significance of eigenflows was raised as an issue in in (?).

According to (?), the difference between the behaviour of social network traffic, which will be considered in this paper, and traditional traffic theory is that it takes in account end-to-end patterns, rather than traffic behaviour over time. The authors of (?) identified the statistical analysis of data traffic measurements obtained from the local area networks of a campus site. This paper showed that there are a number of TCP applications that cause non stationary behaviour in the aggregate traffic stream. These features then cause difficulties for traditional prediction methods.

The challenge for today's traffic engineers is to gain an understanding of the likely characteristics of future traffic behaviour. Understanding the characteristics of traffic has the potential to provide the insight into how traffic behaves that is needed to develop a model of traffic which provides a deeper understanding. In (?), for example, it is asserted that a strong traffic model is required in order to design Internet networks with satisfactory QoS architecture. Estimation of social network behaviour from captured data sets is important because it provides a deeper understanding of the intensity and qualities of communication between the users and servers. Furthermore, modelling and analysis of traffic on all links simultaneously is required in order to solve a wide range of important problems faced by network researchers (?). This could help to get accurate performance evaluation and traffic analyses. In addition, in spite of the importance of the statistical analysis of OD flows, there is little prior work on this field (?).

The authors of (?) observed that there are important characteristics of the underlying social network which can be identified from traffic data obtained from Centre for Applied Internet Data Analysis (CAIDA), without consideration of evolution over time. In this chapter statistics which capture time-independent features of traffic are investigated.

4.3 Traffic Analysis Methods

A number of methods for inferring social network characteristics from traffic data were described and applied to real data in Chapter 3. Some of these methods are further investigated in this chapter, and in addition, some important new methods are considered. In particular, the previous methods largely ignored the investigation of the eigenflows. The eigenflows which are found in (?) were not analysed thoroughly to check whether they are genuine or just occurred by accident. Therefore, in this chapter we have developed a method to address this important question.

The main focus of the work in this chapter is on three types of traffic matrix: the mean, variance and covariance traffic matrices. In this section, the difference between these three types of matrix is explained. Figure 4.2 illustrates the whole “work” and how the three matrices subdivide this into different components. When O–D pairs are independent total work is broken down into only two parts (Mean^2 and Variance). On the other hand, when there is a correlation between the O–D pairs, the total work is divided into Mean^2, Variance and Covariance.

Total traffic (in the sense of $E(X^2)$) can be further subdivided according to its

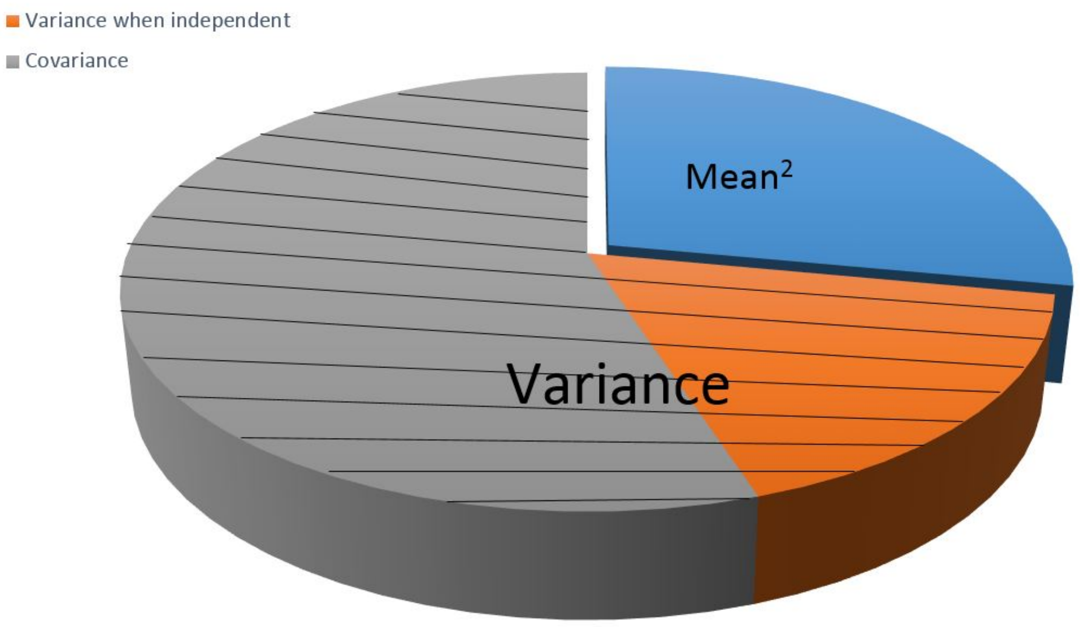


Figure 4.2: Analysis of work $(E(X(t)^2))$

source and destination. This analysis into components enables the underlying social relationships to be identified. For this purpose, first let us consider just two such components.

Suppose X represents the traffic between the nodes i and j at time t, and Y represents the traffic between nodes k and l. The expectation (mean) of the sum of two such quantities satisfies the identity:

$$E(X+Y) = E(X) + E(Y). \tag{4.1}$$

the definition of variance is:

$$\begin{aligned}\mathrm{Var}(X) &= E(X - E(X))^2 \\ &= E(X^2) - 2E(X(E(X))) + E(X)^2 \\ &= E(X^2) - (E(X))^2 \end{aligned} \tag{4.2}$$

If $X \perp Y$, (X is uncorrelated with Y),

$$\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y).$$

If $X \not\perp Y$

$$\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\mathrm{Cov}(X, Y)$$

the detailed components will be: If $X \perp Y$

$$\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) \tag{4.3}$$

If $X \not\perp Y$, on the other hand,

$$\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\mathrm{Cov}(X, Y). \tag{4.4}$$

The rules (4.1), (4.2), (4.3), and (4.4) can be used to break down the traffic into components in a manner which explains their relative importance. This way of analysing data is used often in research in diverse fields under the name *Analysis of Variance*. As the name suggests, normally, the *mean* is not included in the analysis. However, in the case of traffic, there are cases where the mean is more important than the variance. For example, this is true when traffic is highly aggregated. Furthermore, (4.2) shows how the mean can be included in the analysis. Instead of dividing $\mathrm{Var}(X)$ of the traffic, X into different components, we can divide the $E(X^2)$ of the traffic X into components. Since (4.2) shows that $E(X^2) = E(X)^2 + \mathrm{Var}(X)$, the contribution, $E(X)^2$ due to the mean, can be considered as just one of the components of $E(X^2)$.

In the remainder of this chapter, successive refinement of traffic volumes, in the sense of $E(X^2)$, will be conducted. For convenience, the term *work*, will be used as synonymous with $E(X^2)$.

Henceforth, we refer to a certain interval of time by reference to its starting time, which is denoted by t. This interval of time, t to $t + \delta t$, should be thought of as a "typical" interval within the sample, or within the operation of the network. Its characteristics will be inferred from the whole sample, and hence it does not represent any specific period of time, but rather it represents all moments.

The duration of this interval can vary, but is typically 50 – 100ms, which is of the same order as the duration of traffic which can be stored in a buffer. Experiments are conducted below to determine how sensitive the analysis is to the choice of time interval. In fact, buffers will usually be even smaller than this, especially in a core network, where link capacities are higher. When the quantity of traffic X transferred between the nodes i and j at time t, the detailed components will be as follows.

(i) If each O–D pair is completely independent on the other, then:

$$\mathrm{Var}(\sum_{(i,j)=(1,1)}^{N,N} X(t)) = \sum_{(i,j)=(1,1)}^{N,N} VarX_{i,j}(t)$$

(ii) If the covariance between O–D pairs is non-zero, then:

$$\mathrm{Var}(\sum_{(i,j)=(1,1)}^{N,N} X(t)) = \sum_{(i,j)=(1,1)}^{N,N} \mathrm{Var}X_{i,j}(t) + \sum_{\substack{(i,j)=(1,1),(k,l)=(1,1)\\(i,j)\neq(k,l)}}^{(N,N),(N,N)} \mathrm{Cov}_{(i,j),(k,l)}(X_{i,j}(t)X_{k,l})$$

At any time the total work is represented by:

$$\begin{aligned} E\left(X(t)^2\right) &= \sum_{(i,j)=(1,1)}^{N,N} M_{i,j}(t)^2 + \sum_{(i,j)=(1,1)}^{N,N} \mathrm{Var}X_{i,j}(t) \\ &\quad + \sum_{\substack{(i,j)=(1,1),(k,l)=(1,1)\\(i,j)\neq(k,l)}}^{(N,N),(N,N)} \mathrm{Cov}_{(i,j),(k,l)}(X_{i,j}(t)X_{k,l}) \end{aligned} \tag{4.5}$$

Where $\{M_{ij}(t)\}$ is matrix of mean values.

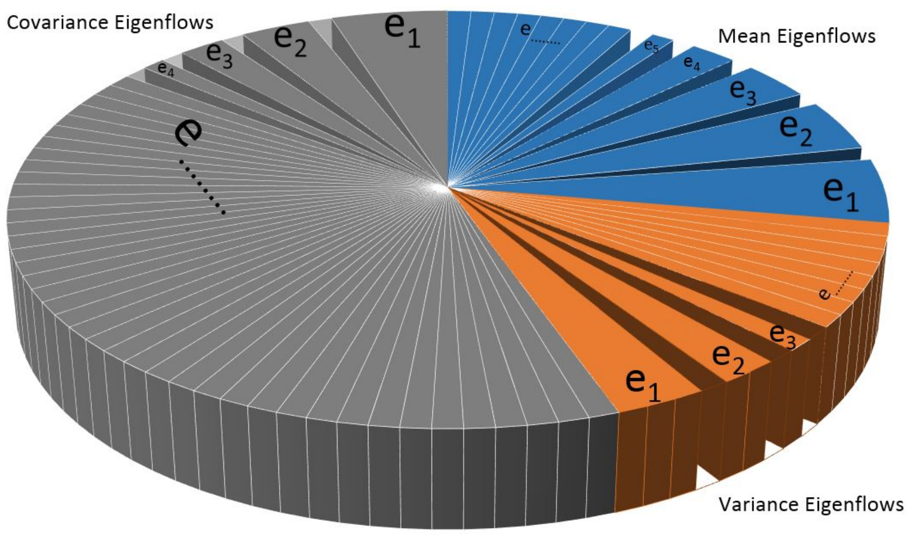


Figure 4.3: Eigenflows distribution for whole work.

4.4 Eigenflow Analysis

In order to identify social features of the traffic, we now further break down the traffic matrices in (4.5) by spectral analysis (eigenvector decomposition). Also, each component in (4.5) is subdivided into a sequence of eigenflows. Figure 4.3 illustrates the way that each analysis method is subdivided to number of eigenflows ($e_1, e_2, e_3, ...$), where $e_1 > e_2 > e_3 > ......$

Consider the mean matrix, $M = (M_{ij})_{i,j=1,1}^{N,N}$. A singular value decomposition of this matrix takes the form:

$$M = U^T DW$$

where U and W are orthogonal (unitary) matrices, and D is diagonal, with its diagonal entries ordered in decreasing order of magnitude. A similar decomposition can be found for the variance matrix, and the covariance matrix. The defining characteristics and details of each of these decompositions is discussed further in Sections 4.4.1, 4.4.2 and 4.4.3, below.

In each case, the matrices U and W define features of the given matrix (eigensources, eigendestinations or eigenflows), and the eigenvalues (entries on the diagonal of D) are the strength of these features. It is conceivable that there are highly important correlations between one O–D flow and another. For example, if a certain group of users engage in communication behaviour simultaneously, as a consequence of some shared activity or goal, this will lead to correlated O-D flows. The Antraff (**?**) software is currently able to undertake singular value decomposition of traffic matrices and also to extract the eigenflows, and to display the O-D flows from which they are composed. However, it remains possible, at this stage, that the eigenflows are actually artificial, i.e. they are not actually a consequence of any genuine correlation between O-D flows.

If eigenflows are significant it can only be due to correlation in time between different O-D pairs. Eigenflows will arise, for example, when Apple, or Google, release an update of their phone operating systems. In the subsequent hours and days these updates will be downloaded by millions of devices. This is not an attack, but just a consequence of the way Internet technology is managed.

4.4.1 Analysis of Mean

The *mean* traffic matrix, M, for the traffic is defined as

$$M = \begin{pmatrix} M_{11}(t) & \dots & M_{1n}(t) \\ \vdots & \ddots & \vdots \\ M_{n1}(t) & \dots & M_{nn}(t) \end{pmatrix} \tag{4.6}$$

Each one of the indices in the above matrix is the *rank* of an origin or a destination. The Antraff software ranks the origins and destinations by the total bytes sent/received to/from the specific node. The rank of a node as an origin will not, usually, be the same as its rank as a destination. The software uses the ranks of origins and destinations because the total number of origins and destinations is much too large to be able to construct matrices which include *all* the IP addresses which occur in the trace.

The *mean* traffic matrix contains the average bytes of the data from the origin to the destination, during the period of time considered. This matrix is estimated from a traffic trace by accumulating the sum of the bytes between each origin and destination (interval by interval) and then dividing this by the number of intervals. We can apply SVD to this matrix directly. A singular value decomposition of this matrix takes the form:

$$M = U^T D W$$

where U and W are orthogonal (unitary) matrices, and D is diagonal, with its diagonal entries ordered in decreasing order of magnitude. These conditions uniquely determine the matrices U, D and W (except under the condition where two or more eigenvalues coincide, which is highly unlikely). Algorithms to determine them are readily available in libraries of software for carrying out computational linear algebra (?).

Figure 4.4 illustrates an interpretation of this singular value decomposition. Each row of the matrix U^T is a linear combination of the original sources (of most importance) in the traffic; likewise, each column of the matrix W represents a linear combination of the original destinations. Let us call these linear combinations *eigensources* and *eigendestinations*. By the nature of a singular value decomposition, the first eigensource and eigendestination are uniquely defined by the fact that from all possible choices, these two are the linear combinations of sources and destinations which fit, in the least-squares sense, the original traffic matrix as closely as possible, or to put this another way, they *explain* the traffic data better than any other

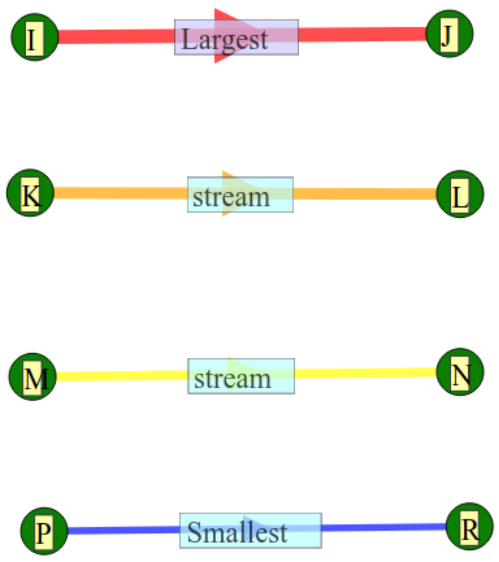


Figure 4.4: An interpretation of the rotation of axes (sources and destinations) of a mean (or variance) traffic matrix.

choice. Similarly, the second eigensource and eigendestination explain the remaining data better than any other choice. And so on, until the entire traffic matrix is explained by the complete list of eigensources and eigendestinations. The volume of traffic explained, by each pair of eigensources and eigendestinations, is given by the corresponding eigenvalue.

The mean traffic matrix can introduce some undesirable smoothing of the traffic. In particular, if the traffic is very bursty, the mean traffic matrix will not record this feature, and relying on it alone will lead to under-provisioning. We can avoid this by using a slightly different matrix which is the Variance matrix.

4.4.2 Analysis of Variance

The Variance matrix, V, for the traffic is defined as

$$V = \begin{pmatrix} V_{11}(t) & \dots & V_{1n}(t) \\ \vdots & \ddots & \vdots \\ V_{n1}(t) & \dots & V_{nn}(t) \end{pmatrix} \tag{4.7}$$

The indices of the *variance* traffic matrix represents, like in *mean* traffic matrix, the rank of an origin or a destination. The nodes are ranked by the total bytes sent/received to/from the specific node. The *Variance* traffic matrix, like mean matrix, is indexed by origins and destinations. In this matrix it is necessary to subdivide the traffic into time-slots during the analysis. The choice of time–slot length is significant, and will be varied. A typical value in the analyses presented here is 0.05 seconds. When forming the *variance* matrix, the *square* of the bytes between each origin and destination, in each interval, is accumulated in each entry of the matrix and divided by the interval length and then the corresponding square of the mean traffic between the same origin and destination is subtracted.

We can also apply SVD to this matrix directly. A singular value decomposition of this matrix takes the form:

$$V = U^T D W$$

the matrices U and W define features of the given matrix (eigensources, eigendestinations), and the eigenvalues (entries on the diagonal of D) are the strength of these features.

Figure 4.4 illustrates, as in *mean* case, an interpretation of this singular value decomposition. The original sources are represented by the rows of the matrix U^T as as a linear combination. Alike, the original destinations are represented by the columns of the matrix W as a linear combination. These linear combinations are called *eigensources* and *eigendestinations*. By the nature of a singular value decomposition, the first eigensource and eigendestination are uniquely defined by the fact that from all possible choices, these two are the linear combinations of sources and destinations which fit, in the least-squares sense, the original traffic matrix as closely as possible, or to put this another way, they *explain* the traffic data better than any other choice. Similarly, the second eigensource and eigendestination explain the remaining data better than any other choice. And so on, until the entire traffic matrix is explained by the complete list of eigensources and eigendestinations. The volume of traffic explained, by each pair of eigensources and eigendestinations, is given by the corresponding eigenvalue.

There are two variants of this type of matrix: if the mean-squared of the traffic in each interval is subtracted from the mean sum-of-squares, we arrive at the traditional variance, of the traffic passing between each origin and destination. However, it is also useful *not* to subtract the mean-squared. This matrix, which we could perhaps term the *power* matrix, includes the mean-squared, and hence is a better measure of the end-to-end load.

4.4.3 Analysis of Covariance

The covariance matrix, C, for the traffic is defined as

$$C = \begin{pmatrix} C_{11}(t) & \dots & C_{1n}(t) \\ \vdots & \ddots & \vdots \\ C_{n1}(t) & \dots & C_{nn}(t) \end{pmatrix} \tag{4.8}$$

where each of i and j is the rank of an O-D pair, and C_{ij} denotes the covariance of the bytes in O-D pair i with the bytes in O-D pair j.

Traditionally principal component analysis decomposes the quantity being analysed into uncorrelated components formed as linear combinations of the original components. Let us suppose the O-D traffics are X_i, $i = 1, \ldots, n$. Thus, a transformed component will take the form $\sum_{i=1}^{n} u_{ki} X_i$, for some vector $u_k = (u_{k1}, \ldots, u_{kn})'$ with $\sum_{i=1}^{n} u_{ki}^2 = 1$. Two such transformed components, with coefficients of vectors u_1 and u_2, will be uncorrelated if $\mathrm{Cov}\left(\sum_{i=1}^{n} u_{1i} X_i, \sum_{i=1}^{n} u_{2i} X_i\right) = 0$. A principal component analysis seeks a complete set of such components such that:

(i) the variance of the first k components is larger or equally large to any other such decomposition, for all $k = 1, \ldots, n$;

(ii) each component is uncorrelated with all the other components.

The total variance explained by this analysis will be the same as any other analysis into independent components. This follows from the invariance of the trace (see (ii) below). The sum of the variance of uncorrelated components of traffic will, in general, be less than $E\left(X(t)^2\right)$. It is conceivable that a different decomposition of traffic in which this quantity is preserved could be developed, based on the invariance of the Frobenius norm (see (i) below), however, this is not the way principal component analysis is traditionally constructed, and so it has not been pursued.

The matrix representation $C = UDU^T$, in which U is orthogonal, U^T denotes the transpose of U, and D is diagonal, with diagonal entries in decreasing order of magnitude, is called a *singular value decomposition* (SVD), of the matrix C. It exists whenever the matrix C is symmetric. It can be applied, in particular, to either the covariance matrix, or the matrix $E\left(X_i X_j\right)$. In both cases the SVD leads immediately to the principal component analysis, as defined above in the sense that the rows of the matrix U are the vectors defining the principal components of the covariance matrix. The variance explained by a singular value decomposition $C = UDU^T$ is the sum of the diagonal elements of D, i.e. $\mathrm{VE} = \sum_k D_{kk}$. In general, $\mathrm{VE} < E(X^2)$. The excess of $E(X^2)$ over VE is due to the fact that X is composed of correlated traffic.

This matrix type, which is the type used in (**?**) is formed by using O-D *pairs*, rather than origins and destinations, as the indices of the rows and columns. In each time-slot, the bytes delivered between each origin and destination are determined, and the *contents* of entry C_{ij} in the covariance traffic matrix for this time-slot is the *product* of the bytes sent between O-D pair i and O-D pair j. These are then averaged over the trace, and the product of the mean traffics for these O-D pairs subtracted, to form the covariance traffic matrix.

Although these three traffic matrices may seem rather different, it turns out that they all reveal similar features of the traffic. The most prominent feature, shared by all the matrices, is the Pareto principle, that a small number of eigenvalues explain a high proportion of the original traffic. In addition, the discrepancy between the eigenvalues of the covariance matrix and those of the variance and mean matrices provides evidence of correlation between O-D traffics.

4.4.4 Analysis Algorithms

The mean, variance, and coveriance of a traffic sample are matrices, and therefore the obvious method of spectral analysis is to carry out a singular value decomposition (SVD) of each of these matrices. The mean and variance matrices are indexed by sources and destinations, and are not symmetrical. The covariance matrix, on the other hand, is indexed by origin-destination pairs (OD pairs), and is symmetrical.

Consequently, the spectral analysis (SVD) of the mean and variance matrices will lead conceptually to a representation of the traffic as flowing from n *eigen-sources* to n *eigen-destinations*, in which each eigen-source and eigen-destination is a mixture of the original sources and destinations, whereas the spectral representation of the covariance matrix will lead conceptually to a representation of the traffic (after subtracting the mean), into n eigenflows, each of which is a mixture of O-D traffics.

In order for these analyses to be compatible, and in order that they can be validated, it will be useful if each method can be viewed as *decomposing* the original traffic into components. That is to say, the traffic components, in each case, should add up to the original traffic. This requirement forces us to carry out the calculations in a specific way.

Singular value decomposition can be explained in the following way: given a matrix of experimental results, A say, we seek a unitary (orthogonal) transformation, also known as a *rotation*, of the axes in which A is expressed, such that A becomes as simple as possible (diagonal), and also so that its features are ordered in importance. Applying a rotation, U, to the row axis of the matrix A, is effected by multiplication on the left: $A' = UA$. Applying a rotation V^T to the column axis of the matrix A is effected by multiplication on the right: $A'' = AV^T$. Combining both, to produce a diagonal matrix, is achieved by multiplication on both sides: $D = UAV^T$. If D is diagonal with the elements on the diagonal ordered in size, this is a *singular value decomposition*. In the special case when A is symmetric, we can restrict our choice of U and V further by requiring that $U = V$.

Invariants of unitary transformation

There are two invariants of matrices under unitary transformation that are relevant to the type of SVD algorithm to apply to traffic:

(i) Invariance of Frobenius (Hilbert-Schmidt) norm. The Frobenius norm (**?**) (also called Hilbert-Schmidt norm) of a matrix, A for example, is the sum of squares of its elements, i.e.

$$| A |_{\mathrm{HS}}^{2} = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}^{2}.$$

If U and V are unitary (orthogonal), $| UA |_{\mathrm{HS}} = | AV^{T} |_{\mathrm{HS}} = | A |_{\mathrm{HS}}$.

(ii) Invariance of trace. For any matrix, A, the trace is defined by

$$\mathrm{Tr}(A) = \sum_{i}^{n} A_{ii}$$

which is also invariant under simultaneous orthogonal transformation of both axes, i.e.

$$\mathrm{Tr}(UAU^{T}) = \mathrm{Tr}(A).$$

This follows from the more general property
$\mathrm{Tr}(UAU^{-1}) = \mathrm{Tr}(A)$, for any invertible matrix U, which follows from (14) in (?).

We can use these invariants to ensure that the particular form of the singular-value decomposition used for each traffic matrix preserves the *quantity of traffic* represented by the matrix. As discussed earlier in this section, quantities of traffic should be measured in bytes^2. This is, indeed, the unit in which variance and covariance matrices are also measured. The quantity we wish to analyse (sub-divide) is $E(X(t)^2)$, where $X(t)$ is the traffic arriving in a certain time interval. This is measured in bytes^2.

4.5 Antraff software

In the previous chapter the Antraff software package has been developed to analyse several datasets with massive data, about tens of millions of packets obtained from CAIDA, and accumulate the traffic matrices by the three different methods, for the mean, variance, and covariance matrices, as explained in Section 3.4. The Antraff package was further developed to carry out the experiments of this chapter. This software is able to calculate the singular value decomposition (SVD) for the three estimated matrices to extract and plot the *eigenflows* of each type of matrix. Furthermore, the software has been designed to extract and plot the *eigenflows* corresponding to the largest three eigenvalues of each type of traffic matrix.

4.6 Experimental results

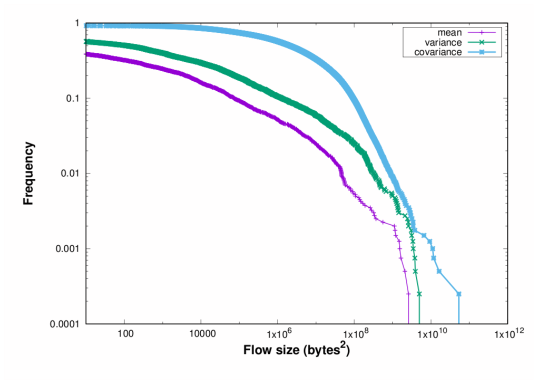
The eigenvalues of the three matrices, mean, variance and covariance, have been analysed in this chapter. Four groups of data are used in the experiments came from different places and different years, all these data–sets obtained from CAIDA. Each set has 10 data files, and each file has about 5 millions packets captured during *one* minute. Thus, the *mean* matrix is considered as short–term. If a mean traffic is measured over a longer period of time it will vary as the length of this period increases.

The mean and covariance matrices have been calculated in different ways which mean that literal comparisons of the bytes^2 should not be relied upon. In particular, the mean matrix is calculated by sampling the largest origins and destinations, whereas the covariance is calculated by sampling the largest O–D pairs. Typically, this means the covariance is limited to 4000 O–D pairs, whereas the mean matrix is limited to 4000×4000 O–D pairs. Hence, the amount of data which has been ommitted from the analysis, because only the most important origins, destinations, and O-D pairs can be included, is greater for the covariance matrix than for the man matrix. Also, we deliberately squared the eigenvalues of mean matrix to be consistent with the eigenvalues of variance and covariance matrices–i.e. so that they are all measured in bytes^2. There is another difference between the way the mean curve in Figure 4.6 is calculated and the variance and covariance curves in this figure are calculated which is that the components of the mean matrix are summed and *then* they are squared. whereas the components of the variance and covariance matrices, which are already measured in bytes^2, are simply summed. These differences mean that we should not interpret comparisons between the proportion of total bytes^2 *explained* by each matrix too literally.

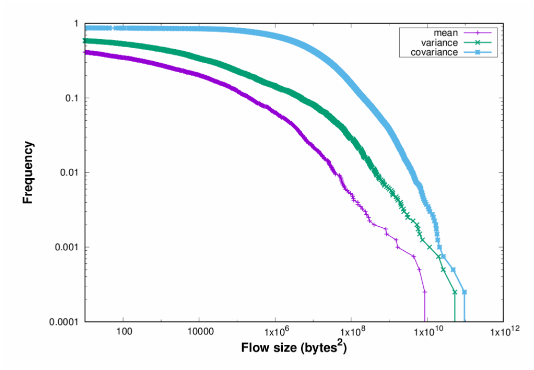
Figures 4.8–4.10 show the eigenvalues of the three different types of traffic matrices which were investigated in different ways from the four datasets with 0.05 second sampling interval. All the experiments display the Pareto principle, which is that a small proportion of eigenflows are associated with a high proportion of the traffic. On the other hand, the analysis of covariance traffic matrix suggests that there is a correlation between different flows in the social networks traffic.

To check the effect of changing the sampling interval, the experiments have been repeated with two other sampling intervals: 0.1 and 0.02 seconds, as shown in Figures 4.5–4.7 and 4.11–4.13. The experiments with shorter and longer sampling intervals exhibit very similar behaviour, from which we can conclude that the choice of sampling interval does not have a strong influence on the results. Calculating the traffic matrices from the traces is significantly slower with shorter sampling intervals, so the invariance of the results to choice of sampling interval suggests that a compromise choice, where the interval is somewhat longer than a typical buffer size, may be satisfactory.

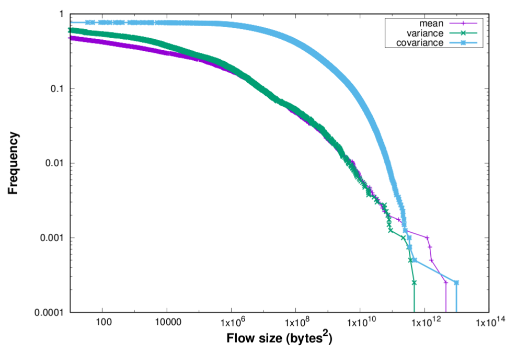
Figure 4.5 shows the actual values of eigenvalues of the mean, variance and covariance traffic matrices in bytes^2. These values (titled ‘flow size’ in the plot) have been plotted on the x axes against the proportion of eigenvalues on the y axes. The eigenvalues of the covariance traffic matrix are larger than mean and variance eigenvalues. This suggests, on the one hand, that there is correlation between different flows in the social networks traffic. On the other hand, this figure illustrates the Pareto principle.



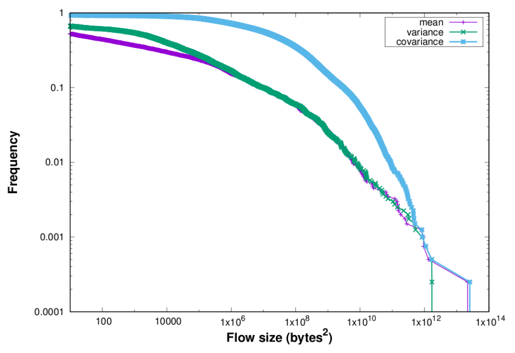
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set



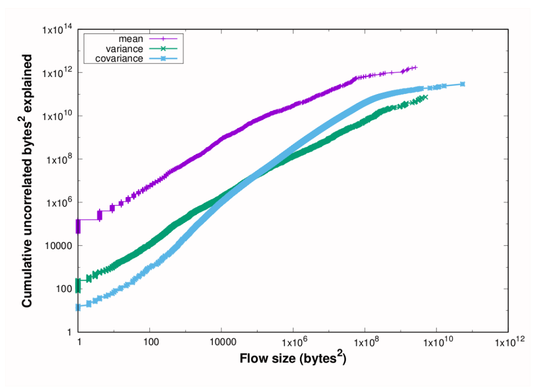
(c) Trace used: Chicago 2015 data-set



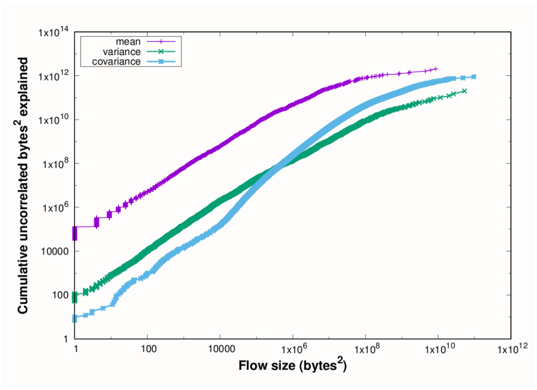
(d) Trace used: Chicago 2016 data-set

Figure 4.5: Eigenvalues of the mean, variance and covariance traffic matrices for four different data sets with 0.1 sec sampling interval time.

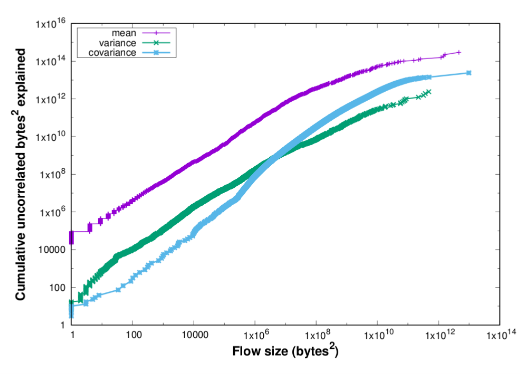
In Figure 4.6, the cumulative proportion of bytes^2 explained by eigenflows up to a certain flow size is plotted against the flow size in bytes^2. This figure is plotted on a log–log scale. It is considered desirable to make clear comparison of importance between the three methods; mean, variance and covariance in terms of the largest individual eigenvalues. The relative height of the covariance and variance curves suggests that the total bytes^2 explained by the covariance matrix is greater than that explained by the variance matrix. However, as discussed above, this comparison is not applicable between the mean and the other methods because the mean has been calculated in different way. For the three methods, the figure represents, as discovered in Chapter 3, that the social network traffic exhibits Pareto principle. Most traffic of each matrix is explained by small number of eigenflows.



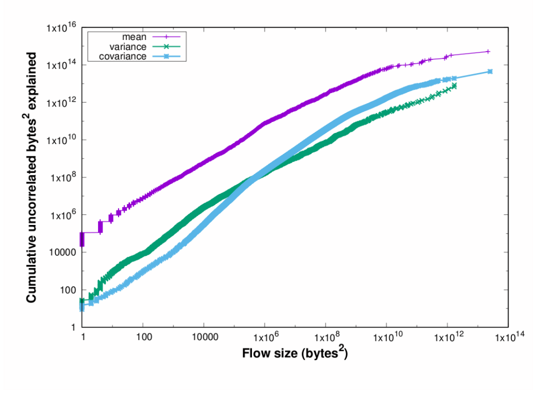
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set



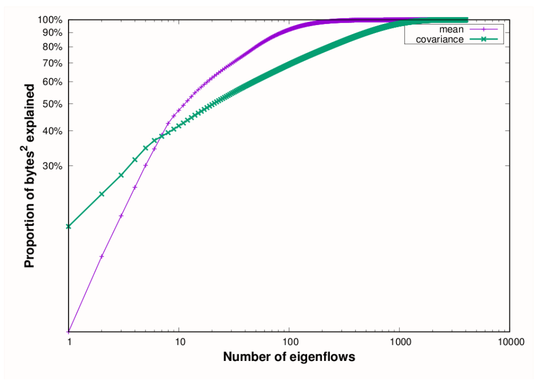
(c) Trace used: Chicago 2015 data-set



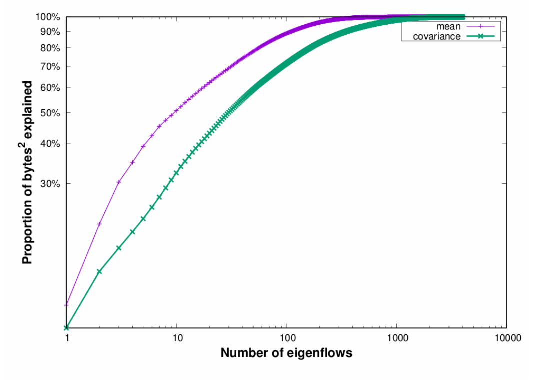
(d) Trace used: Chicago 2016 data-set

Figure 4.6: Cumulative proportion of flows vs flow size in bytes^2 with 0.1 sec sampling interval

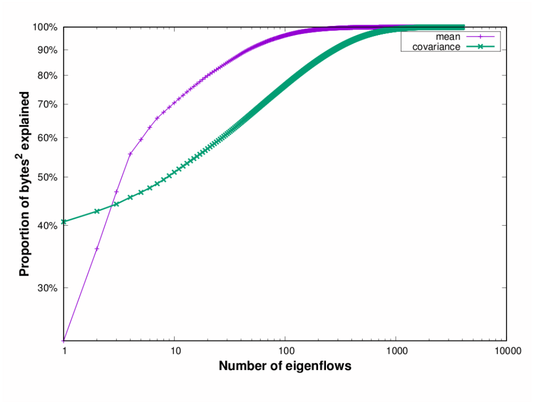
Figure 4.7 illustrates the proportion of total bytes^2 explained by the mean, or covariance matrices (respectively), explained by the largest k eigenflows, of each case, against k. This figure, again, shows the very strong Pareto property of the eigenflows. The O–D pair eigenflows strongly exhibit the Pareto principle. In Chicago 2014 data-set, for example, the first 100 eigenflows, which is 2.5% from all eigenflows of covariance traffic matrix, explains about 75% of the traffic, in bytes^2, of covariance matrix. This is true for all other data-sets.



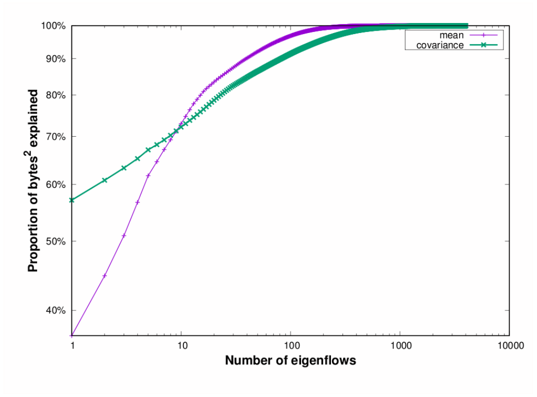
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set



(c) Trace used: Chicago 2015 data-set



(d) Trace used: Chicago 2016 data-set

Figure 4.7: The proportion of bytes^2 explained by every eigenflow with 0.1 sec sampling interval

4.6.1 Estimating γ

Another way to show that the traffic exhibits the Pareto principle is by calculating γ values (sometimes referred to as α), i.e. half the slope of the frequency vs eigenvalue curves on a log-log scale. When a complimentary Pareto distribution with shape parameter (γ) is plotted on a log-log scale it appears as a straight line with slope -γ. Using this fact as an estimation procedure, if we plot the complimentary frequency vs flow-size on a log-log scale, and a significant part of the resulting plot appears to have a linear shape, we can estimate the shape parameter of a fitted Pareto distribution as the negative of the slope of this part of the curve. Halving the slope is necessary because eigenvalues in all these curves are measured in bytes^2.

Tables 4.1–4.3 contain the values of γ for all the curves of the actual values of eigenvalues of the mean, variance and covariance. The slope of most of these curves is about *two*. The smaller the value of γ, the more extreme is the Pareto principle for the case under consideration. In effect, γ is a measure of how *heavy* are the tails of the Pareto distribution in this case, or, to put it another way, how *heavily* the Pareto principle applies. Values of γ about 2 are significant, and values close to, or less than 1, are extreme. This approach to estimating γ, of the traffic, provides a statistically robust method, since it does not rely on identifying flows in the traffic (e.g. by identifying them with tcp flows).

Table 4.1: The values of γ for *Mean* matrix

Data-set	Sampling interval in seconds	γ value (slope of curve)
CAIDA-Chicago-2014	0.1	2.412706
CAIDA-Chicago-2014	0.05	2.23368
CAIDA-Chicago-2014	0.02	2.21621
CAIDA-San Jose-2014	0.1	2.376993
CAIDA-San Jose-2014	0.05	2.267535
CAIDA-San Jose-2014	0.02	2.167537
CAIDA-Chicago-2015	0.1	2.062154
CAIDA-Chicago-2015	0.05	2.068914
CAIDA-Chicago-2015	0.02	2.0341372
CAIDA-Chicago-2016	0.1	2.103812
CAIDA-Chicago-2016	0.05	2.101676
CAIDA-Chicago-2016	0.02	2.017618
Simulation data-file	0.1	1.1158198
Simulation data-file	0.05	1.2024699
Simulation data-file	0.02	1.20224706

4.6.2 Simulation experiments

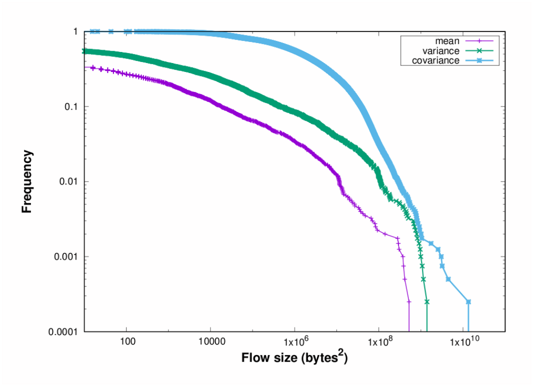
To test the significance of covariance eigenflows which were found in Section 4.6, simulations were constructed using Netml (**?**). Figure 4.14 illustrates the simulated network. The simulations serve as a null hypothesis in which O–D pairs are independent. Since the traces from the simulation produce significantly different results

Table 4.2: The values of γ for *Variance* matrix

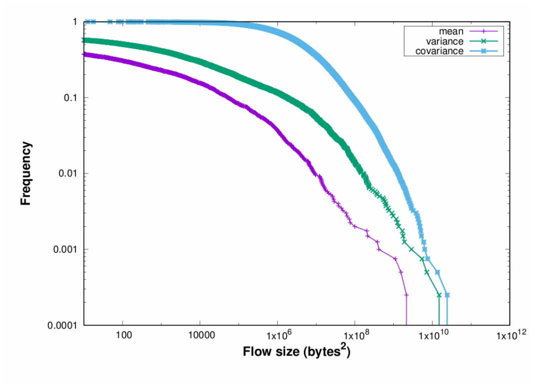
Data-set	Sampling interval in seconds	γ value (slope of curve)
CAIDA-Chicago-2014	0.1	2.043524
CAIDA-Chicago-2014	0.05	2.065645
CAIDA-Chicago-2014	0.02	2.008121
CAIDA-San Jose-2014	0.1	2.0088262
CAIDA-San Jose-2014	0.05	1.9122967
CAIDA-San Jose-2014	0.02	2.0571821
CAIDA-Chicago-2015	0.1	1.996582
CAIDA-Chicago-2015	0.05	1.9176513
CAIDA-Chicago-2015	0.02	2.0029388
CAIDA-Chicago-2016	0.1	2.0048541
CAIDA-Chicago-2016	0.05	2.004863
CAIDA-Chicago-2016	0.02	2.0010523
Simulation data-file	0.1	1.0995108
Simulation data-file	0.05	1.2824699
Simulation data-file	0.02	1.2990365

Table 4.3: The values of γ for *Covariance* matrix

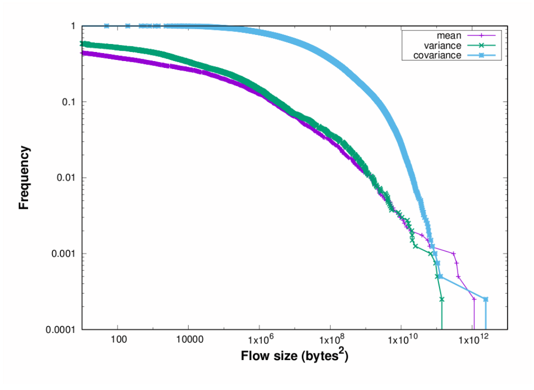
Data-set	Sampling interval in seconds	γ value (slope of curve)
CAIDA-Chicago-2014	0.1	2.06704
CAIDA-Chicago-2014	0.05	2.212742
CAIDA-Chicago-2014	0.02	2.135198
CAIDA-San Jose-2014	0.1	2.151074
CAIDA-San Jose-2014	0.05	2.1199947
CAIDA-San Jose-2014	0.02	2.267537
CAIDA-Chicago-2015	0.1	2.0099599
CAIDA-Chicago-2015	0.05	2.193341
CAIDA-Chicago-2015	0.02	2.75575
CAIDA-Chicago-2016	0.1	2.1305209
CAIDA-Chicago-2016	0.05	2.0130803
CAIDA-Chicago-2016	0.02	2.133176
Simulation data-file	0.1	1.2041695
Simulation data-file	0.05	1.1899748
Simulation data-file	0.02	1.1192617



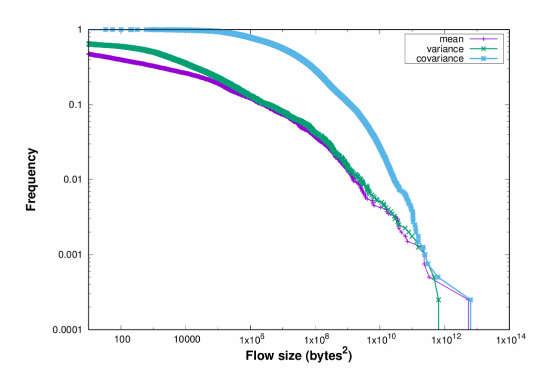
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

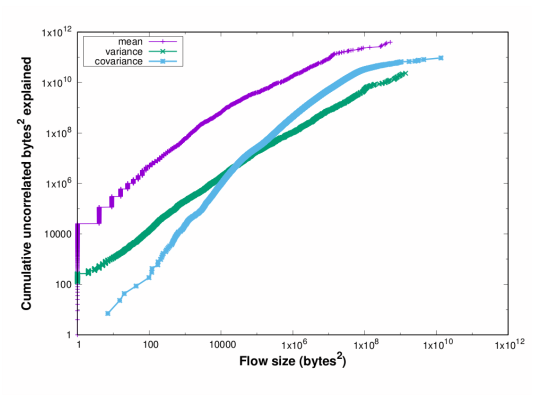


(c) Trace used: Chicago 2015 data-set

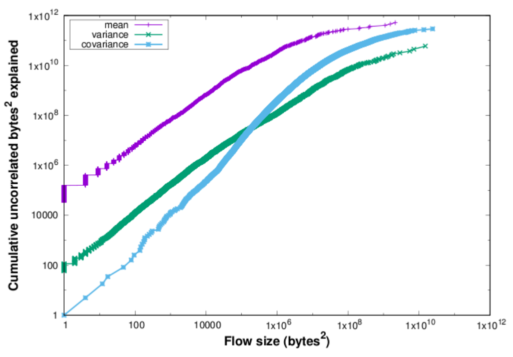


(d) Trace used: Chicago 2016 data-set

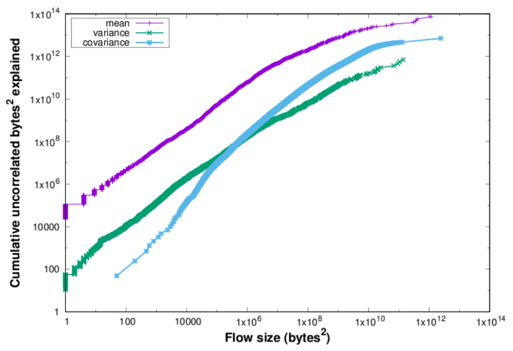
Figure 4.8: Eigenvalues of the mean, variance and covariance traffic matrices for four different data sets with 0.05 sec sampling interval time.



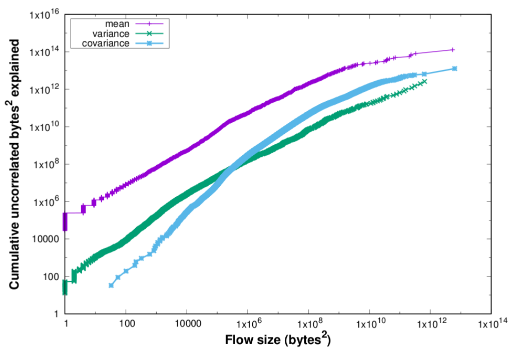
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

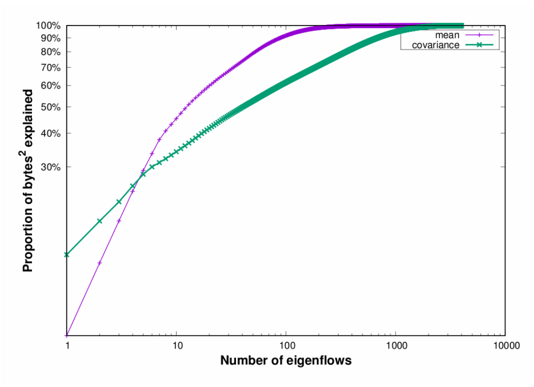


(c) Trace used: Chicago 2015 data-set

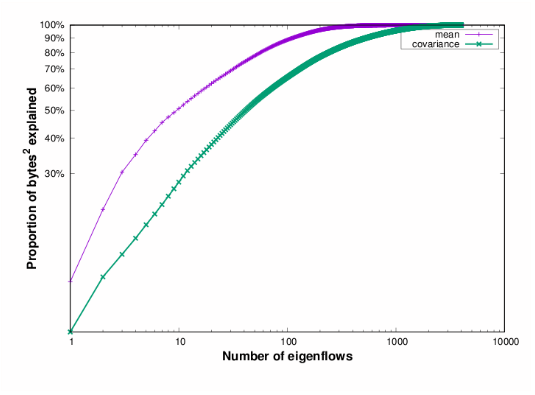


(d) Trace used: Chicago 2016 data-set

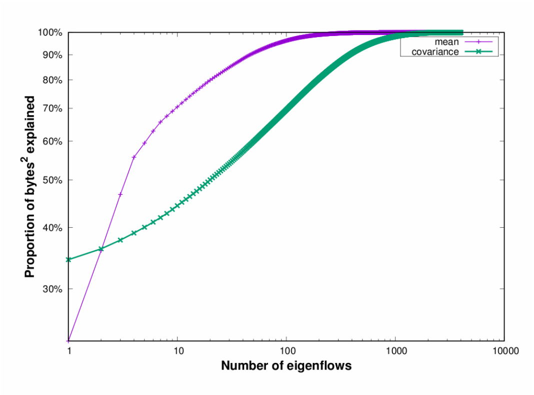
Figure 4.9: Cumulative proportion of flows vs flow size in bytes^2 with 0.05 sec sampling interval



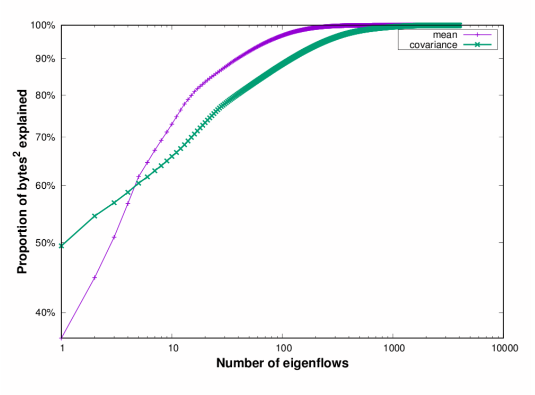
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

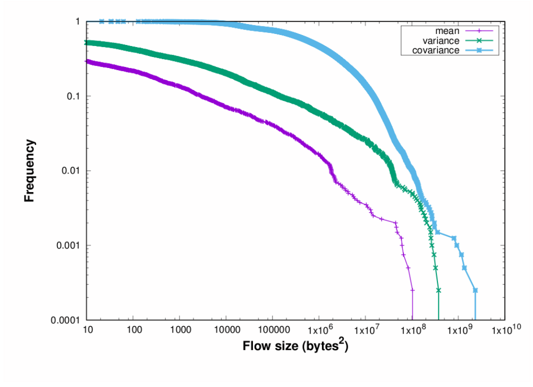


(c) Trace used: Chicago 2015 data-set

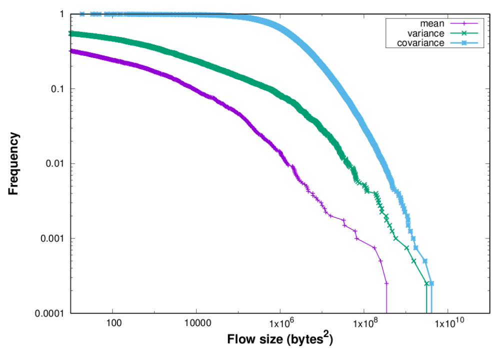


(d) Trace used: Chicago 2016 data-set

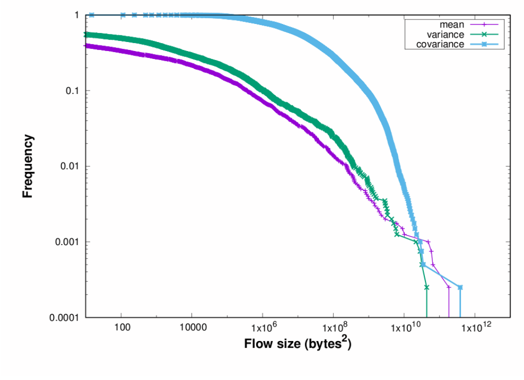
Figure 4.10: The proportion of bytes^2 explained by every eigenflow with 0.05 sec sampling interval



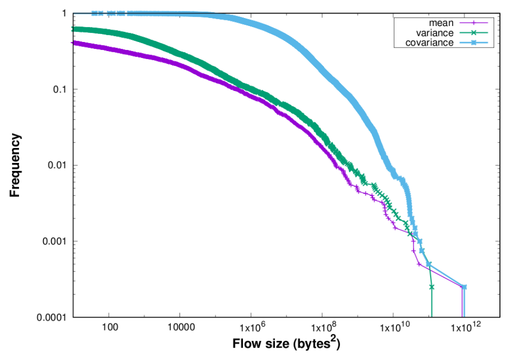
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

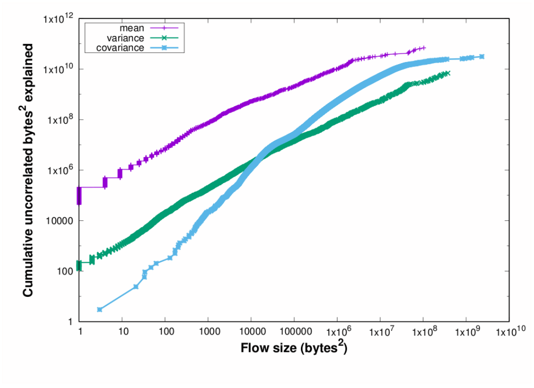


(c) Trace used: Chicago 2015 data-set

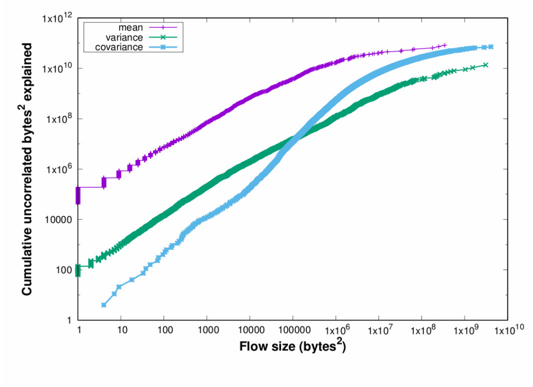


(d) Trace used: Chicago 2016 data-set

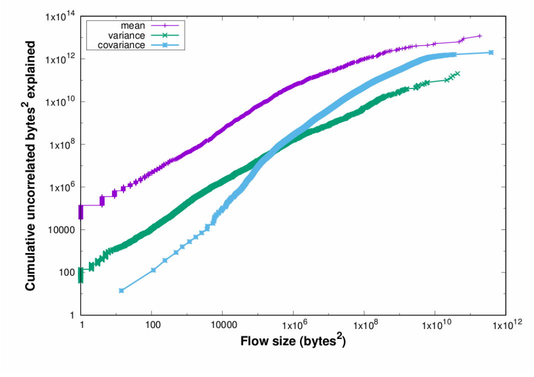
Figure 4.11: Eigenvalues of the mean, variance and covariance traffic matrices for four different data sets with 0.02 sec sampling interval time.



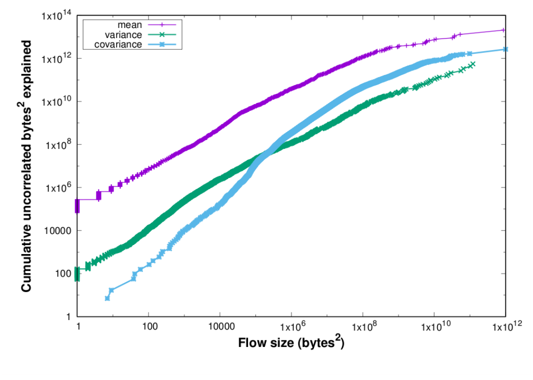
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set

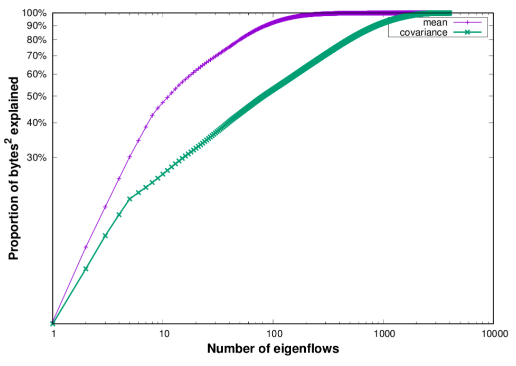


(c) Trace used: Chicago 2015 data-set

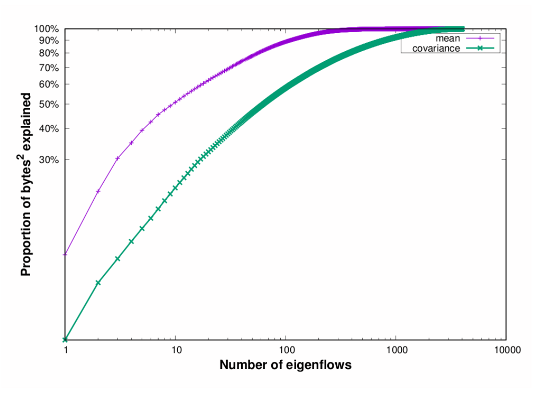


(d) Trace used: Chicago 2016 data-set

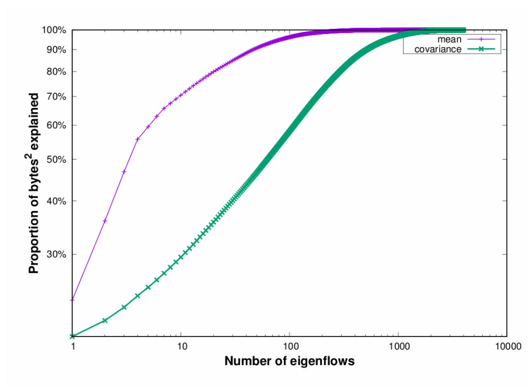
Figure 4.12: Cumulative proportion of flows vs flow size in bytes^2 with 0.02 sec sampling interval



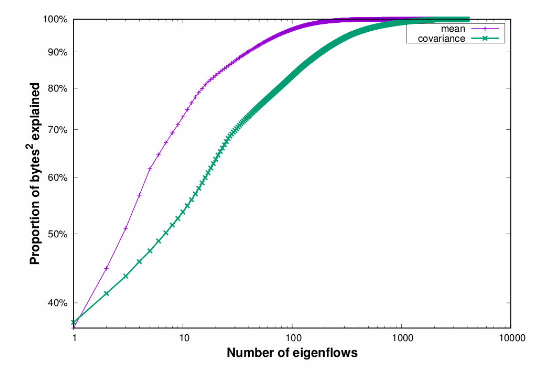
(a) Trace used: Chicago 2014 data-set



(b) Trace used: San Jose 2014 data-set



(c) Trace used: Chicago 2015 data-set



(d) Trace used: Chicago 2016 data-set

Figure 4.13: The proportion of bytes^2 explained by every eigenflow with 0.02 sec sampling interval

when analysed, we can reject the hypothesis that the O-D pairs are independent.The network was constructed with independently distributed O–D traffics of on-off type, with Pareto distributed on periods with $\gamma = 1.2$ and Pareto distributed off periods also with $\gamma = 1.2$. Each link has a capacity of 100 Mbps. In this network traffic model, all traffic stream were statistically independent from each other. The simulation produces a trace file which has been analysed by Antraff, using the same methods as used above on the CAIDA data.

Figure 4.15 shows the simulation result of three different types of analysis: (a) the eigenvalues of the mean, variance and covariance traffic matrices against their position in the list of eigenvalues, (b) the cumulative proportion of bytes^2 explained by eigenflows up to a certain flow size against the flow size in bytes^2 and (c) the proportion of total bytes^2 explained by the mean, or covariance matrices (respectively), explained by the largest k eigenflows, of each case, against k.

Under this hypothesis we find that the results are significantly different from the behaviour observed from CAIDA traces. Figure 4.16 shows that it does not exhibit that the covariance eigenvalues are larger than those from the variance matrix, in fact the covariance and variance eigenflows are nearly identical. This confirms that the O–D flows in the CAIDA traces are not statistically independent, i.e. the correlation between the different flows in the CAIDA traffic is significantly different from zero. Thus, this experiment appears to show that the covariance eigenflows are significant.

In the simulation results γ was estimated in the same way used in CAIDA traces. The γ values estimated from the simulation results, as shown in Tables 4.1–4.3, are similar to the values used for γ in Pareto distributions used to simulate the bursts.

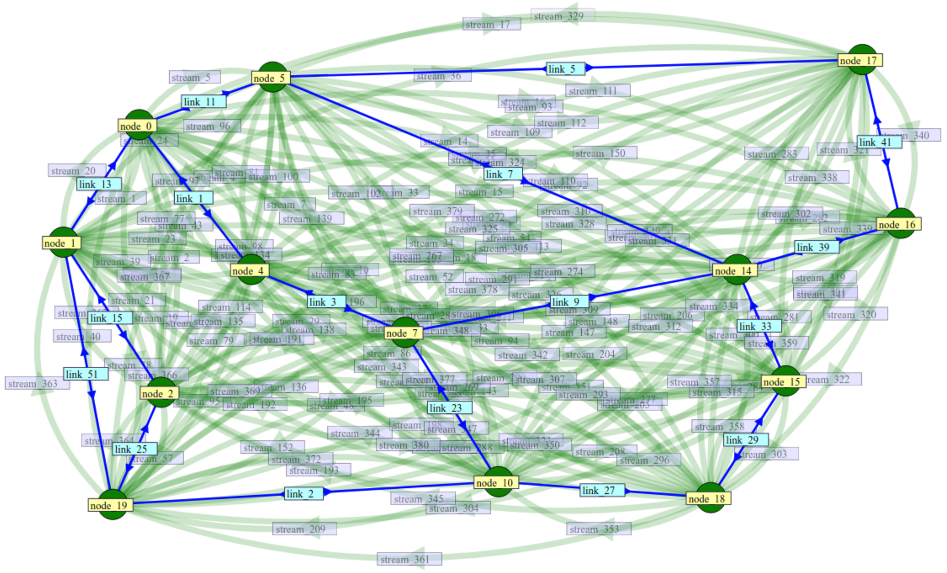
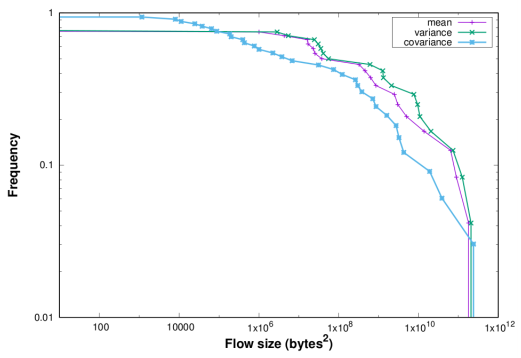
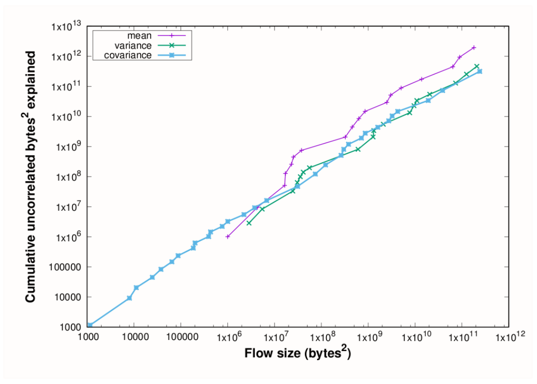


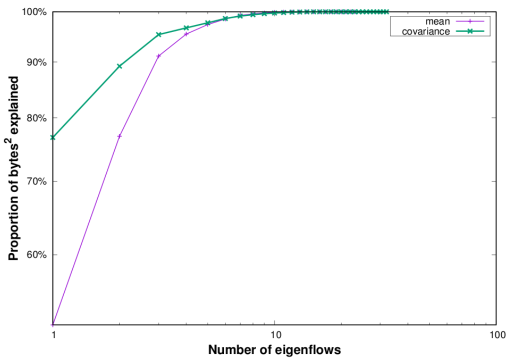
Figure 4.14: Simulated Network



(a) Eigenvalues of the mean, variance and covariance traffic matrices.



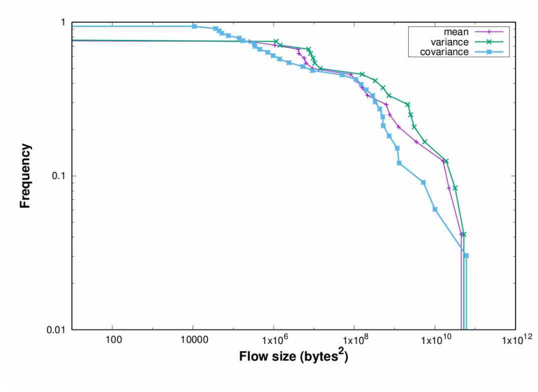
(b) Cumulative proportion of flows vs flow size in bytes^2



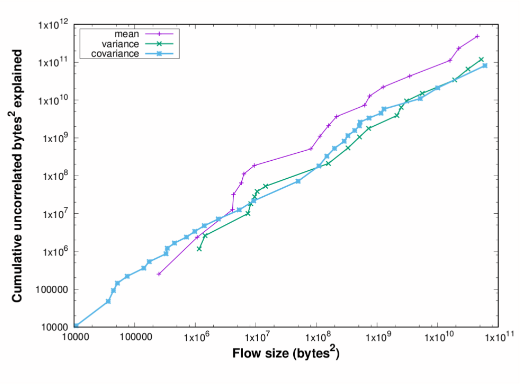
(c) The proportion of bytes^2 explained by every eigenflow.

Figure 4.15: The simulation experiments with 0.1 sec sampling interval

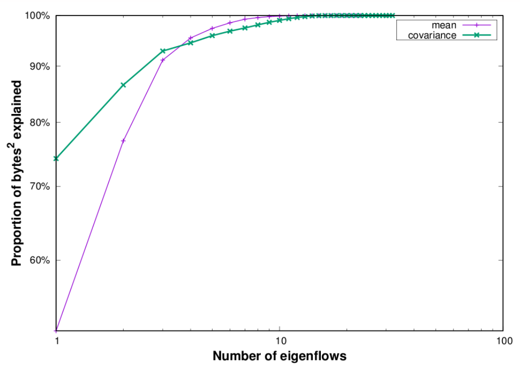
The simulation experiments have been repeated with two different sampling intervals: 0.1 and 0.02 seconds, as shown in Figures 4.15 and 4.17. The simulation results with different sampling intervals, as in CAIDA case, were similar. This means that the sampling interval does not have a strong influence on the results. These experiments also confirm the hypothesis that the slope of the eigenvalue curve can be used to estimate the Pareto shape parameter of the flow sizes in the traffic.



(a) Eigenvalues of the mean, variance and covariance traffic matrices.

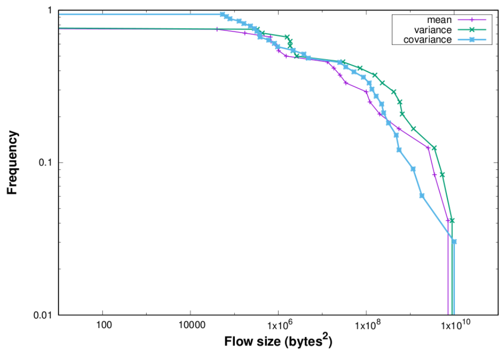


(b) Cumulative proportion of flows vs flow size in bytes^2

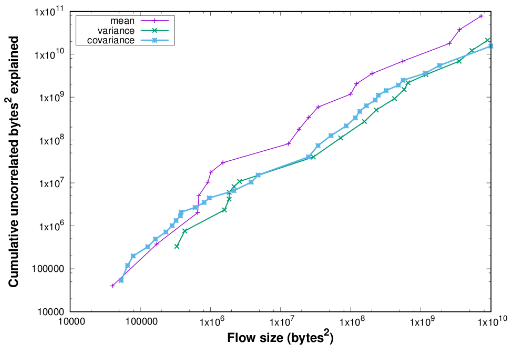


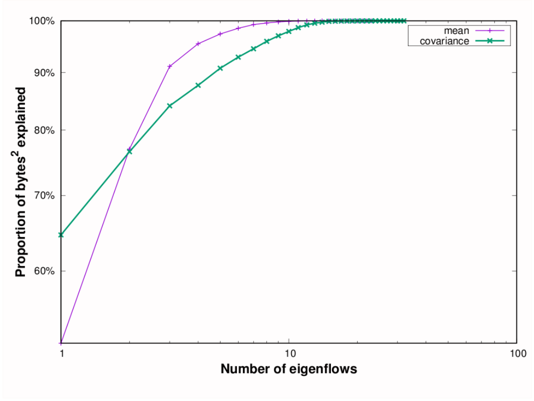
(c) The proportion of bytes^2 explained by every eigenflow.

Figure 4.16: The simulation experiments with 0.05 sec sampling interval



(a) Eigenvalues of the mean, variance and covariance traffic matrices.





(b) Cumulative proportion of flows vs flow size in bytes^2

(c) The proportion of bytes^2 explained by every eigenflow.

Figure 4.17: The simulation experiments with 0.02 sec sampling interval

4.7 Summary of the chapter

Singular value decomposition of traffic matrices of several different types was used to identify *eigenflows* of different types. The significant difference between the eigenvalues in the singular value decomposition of the variance relative to that of the covariance suggests strongly that the covariance eigenvalues are not artefacts, but represent genuine and important correlations between different O–D traffics. A statistical test of significance of the eigenvalues of the singular value decomposition of traffic matrices has been developed by constructing simulations using Netml/ns3. The simulations serve as a null hypothesis. The slope of the frequency vs eigenvalue curves on a log-log scale (γ) has been calculated to show how heavy are the tails of the Pareto distribution in each case.

Chapter 5

Better Service from Understanding Social Network Behaviour

In this chapter we describe and test new approach to better service from social network behaviour understanding. This method is complementary to DiffServ. We call this new method Defensive Services (DefServ) because rather than focussing on differentiation based on previously nominated preferences, it focuses on protection of normal traffic, based on observation and detection of unusual traffic. It is uses a new queue discipline called mice and elephants (MAE).

5.1 Introduction

With the next generation of Internet the Quality of Service (QoS) requirements raised by a wide range of communication–intensive, real–time multimedia applications, best-effort service which does not vary according to conditions or traffic type may not be adequate. This is the sort of reasoning which has lead to the development of the Differentiated Services Architecture, which seeks to define a cost-effective way to ensure QoS in the Internet. DiffServ has been deployed and a modest proportion of traffic in the Internet now carries non-trivial values in the DSCP field of the IP header which will, in some situations, significantly affect performance. However, it is still early days in the deployment of DiffServ and the specific details of how it is deployed vary considerably from operator to operator. Also, the settings which apply to different traffic classes will need to be carefully monitored and studied because otherwise there is a risk of unintended side-effects.

In this chapter we describe an approach to quality of service which is complementary to DiffServ and addresses these issues in quite a different manner. For many years now it has been known that traffic may be broadly classified into two categories: *mice* and *elephants*. Roughly speaking, most bytes are in elephants (large flows), but most flows are mice. The implications of this fact for quality of service are fundamental. In Chapters 3 and 4, we discovered that the social network traffic exhibits Pareto principle. The Pareto principle, which was observed in the CAIDA traffic, is itself evidence of the presence of elephants. In fact, the Pareto principle is in an alternative description of the mice and elephants phenomenon. According to (**?**), detecting and analysing this kind of Internet traffic classification , *mice* and *elephants*, is important significantly to gain a better understanding of Internet traffic behaviour. Protection against the impact of anomalous traffic (elephants), within the Best Effort class or any other class, on any other traffic, will be a major focus in this chapter. Furthermore, although the arrival rate of elephants is relatively low, the presence of elephants, which significantly impact all other traffic, is completely normal and it is important that networks continue to operate satisfactorily despite their presence.

In this chapter a QoS architecture with similar objectives to DiffServ is introduced. We refer to this new QoS architecture as DefServ. It is quite different to DiffServ, in that instead of providing protection for certain privileged classes, it attempts to defend *all traffic* (whatever its class) against the influence of any traffic, when it exists, which is consuming an excess of resources. DiffServ and Defserv have different goals, and hence, can be deployed in concert. Whereas DiffServ emphasises the distinctions between different classes, DefServ emphasises the distinction between different situations.

It has been known for decades in the literature of queueing theory that the when jobs are served according to shortest-job-first (SJF) queue discpline average waiting time for jobs of *all* sizes can (and often does) improve. This is so readily understood by users that it is often introduced spontaneously by customers with large jobs inviting others with short jobs to go first. The benefit of SJF is greater when the distribution of flow sizes is heavy tailed, i.e. the range of job sizes to be handled is greater, and although the frequency of very large jobs is small, their size makes up for it. We now know that this is precisely the case for the distribution of flow sizes in the Internet as explored in Chapter 4. Although router operations focus on *packets*, when a router is modelled as a queueing system, the *jobs* correspond to the *flows*, which are made up of many packets.

If it seems implausible that the performance experienced by flows of *all* sizes can be improved by the simple strategy of serving short flows first, consider this: if the range of sizes is sufficiently heavy-tailed, the benefit of going ahead of larger flows is greater than the disadvantage of waiting for shorter jobs for all jobs, no matter how long the amount of work to implement the required distinction between jobs, is minimal. Furthermore, in Chapter 4, it was shown that the presence of large flows is detectable as a statistical phenomenon, which is therefore *robust*. By identifying the elephants by means of their statistical characteristics, rather than technical features defined in terms of protocol or port, the risk of mis-identification is reduced. Another frequent concern is that implementing SJF may be very difficult, or practically infeasible.

However, the tail-heaviness of sizes helps here also. If the distribution of flow-sizes is sufficiently heavy-tailed, much of the benefit of SJF can be achieved with a highly approximate implementation, in which jobs are de-prioritised only when they exceed a certain size – in fact, only when they have *already* exceeded a certain quite large size. As a consequence, the amount of work to implement the required distinction between jobs, is minimal.

5.2 Related work

The paper (**?**) reports that there are many applications which require service differentiation for satisfactory user quality and, in some cases, without it, the application might not perform at all. For example, companies that use the Internet to perform banking may pay more for better service.

According to (**?**), DiffServ model has been designed by IETF to specify a simple, scalable and coarse-grained mechanism for classifying and managing network traffic for providing QoS in modern IP networks. DiffServ can, for example, be used to provide low-latency to critical network traffic such as voice or streaming media while providing simple best-effort service to non-critical services such as web traffic or file transfers. Moreover in (**?**), the author highlighted that the primary goal of DiffServ was to provide the benefits of QoS without the scalability limitations of IntServ. On the other hand, There are several studies on the performance issues for DiffServ architecture, such as (**?**), claiming that there is a significant influence on Best Effort (BE) class traffic by premium class traffic.

Additionally, the paper (**?**) is concerned with the performance experienced by the *best effort* class of traffic in a DiffServ network. This may seem to conflict with the intent of the DiffServ architecture. The *best effort* class of traffic is meant to receive only as good a quality of service as the network can provide, surely, so *why would we be concerned about the performance* it *receives?*

The paper (?) shows that there is a significant impact on the BE class traffic, as well as other classes, from the routing algorithm used by the premium class service. The authors suggest that the majority of traffic classes in DiffServ model are influenced by one class which has high priority. So it is important to protect the majority of traffic. *This is the same issue highlighted in (?).*

According to (?) the type of network traffic with the greatest volume is the BE class which has the lowest "priority". Other classes take precedence over it and therefore will get better network service. A multi-class routing (MCR) algorithm suggested in (?) was shown to greatly improve the performance of the best-effort class of traffic. The authors of (?) claim that high priority traffic should be transmitted in an efficient manner with the consequent negative influences on lower priority traffic. *Notice how this directly contradicts (?).*

The papers (?), (?), (?), establish that link traffic is long-range dependent. Currently, no whole-of-network traffic model exists. However, there is a significant literature on anomaly detection, (?), and the traffic model implicitly adopted in this literature has the potential to serve in this capacity. They propose to analyse traffic data by Principle Component Analysis (PCA). The dominant few principle components are regarded as anomalies and treated as attacks, or intrusions, but another possibility that should be considered is that there will always be such components and they are not necessarily attacks, but just a consequence of normal human behaviour.

5.3 The Pareto Principle

According to (?), more than a hundred years ago the Italian economist and sociologist Vilfredo Pareto discovered the principle that 20% of the population owned 80% of the property in Italy. The Pareto Principle(?) means that for many phenomena 80% of the results are occurred by 20% of the causes. It is often used in management, economics and business to improve the marketing and make better plan.

After that, it was discovered that the same principle can be applied to the computer science (??). In (?), many Internet traces, which are obtained from very well respected source of Center for Applied Internet Data Analysis (CAIDA) (?), have been thoroughly investigated and it has been discovered that the distribution of flow sizes exhibits the Pareto principle in several ways. Most users contribute very few bytes, and most bytes are from a very small proportion of users. In (?) it is claimed that the Pareto distribution is a good model for Internet flow sizes, rather than the exponential distribution (which was traditionally used for call lengths in telephone networks).

5.4 Netml

Netml (?) is an XML-based language for analysis and development of networks which provides a collaborative working environment for researchers and students of networking. The Netml system can generate an ns-3 program which simulates a network.

The authors of (?) stated that the simulation has been a valuable tool for experimentation and validation of models, architectures and mechanisms in the field of networking. In some of the Internet architectures, (?) simulation is even more valuable, due to the fact that an analytical approach of mechanisms and services is infeasible. There are several notable computer simulation software. Ns-3 is one of these tools. It is a discrete-event network simulator, targeted primarily for research and educational use.

The simulations used in this research were constructed using the Netml system to create an ns-3 programs and run it, and from which the results were then further processed. Figure 5.1 shows the network which is simulated. The key feature in this network, that a collection of flows are competing for access to a congested link, will always occur in networks where there is congestion. Hence, although networks will in general have a great variety of topologies which are quite different to the network shown in Figure 5.1, the results of the simulations are relevant to all these situations.

In this network, all priority 0,1,2 traffic are going to nodes Y0, Y1, Y2 respectively. All of the traffic comes from nodes A – E to their respective destinations through the shared link between node X and Y. The red traffic stream from A to Y0 has been introduced to model the fact that we are deliberately focussing on situations where congestion is present. In some of the simulations conducted, a large flow (an elephant), is modeled by this traffic stream. In others, this flow is not large (so that both situations can be considered. Since we are mainly interested in situations where a large flow is present, we deliberately introduce this flow. This is not to suggest that such a flow will *always* be present, but merely that it is mainly *when* such a flow is present, that we wish to observe the behaviour of the network.

5.5 Network scenarios

To objectively evaluate the performance of any method which aims to improve the QoS in the Internet, we need to ensure that the method is effective in a full range

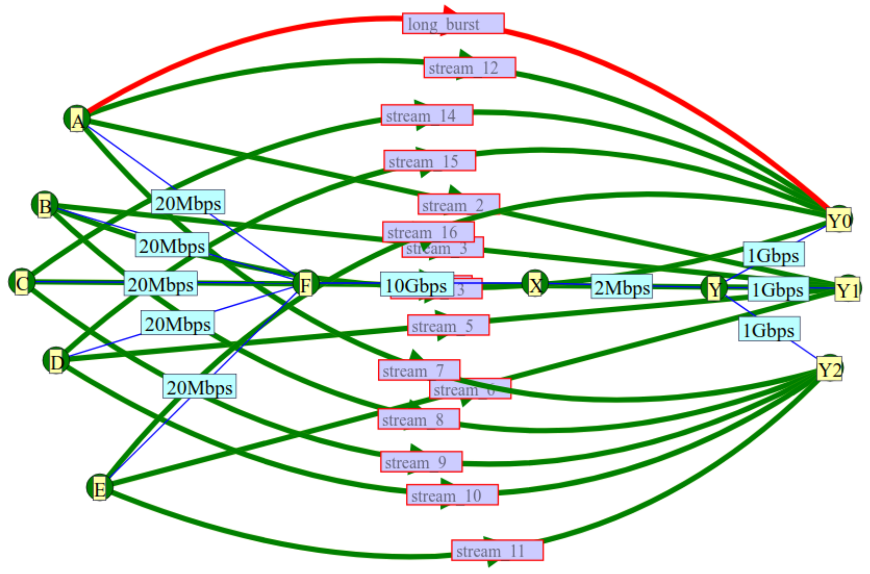


Figure 5.1: The simulated network created on Netml.

of typical network and traffic scenarios. The possible scenarios in our case, as illustrated in Figure 5.2, are: (a) the queue discipline which is used in the network: Diffserv, Defserv or Droptail; (b) whether the network is correctly designed or not; (c) if the network is overloaded or not. A network is correctly designed when the link capacity is sufficient to carry the traffic successfully. The most important scenarios are where the network is temporarily overloaded, because when there is no congestion network performance will usually be good for all users. In other words, when evaluating strategies for handling congestion, it is appropriate to focus on scenarios where congestion is taking place.

To be objective about possible traffic configurations, we can draw on recently gathered knowledge about Internet traffic, as discussed in the previous section. In particular, each traffic scenario can be summarised in terms of the presence or absence of large flows. For example, an important scenario which occurs quite often is where there are no large flows present, of any DiffServ class. Although this scenario is likely to occur, at least locally (i.e. at a certain link), it is one which can be dealt with

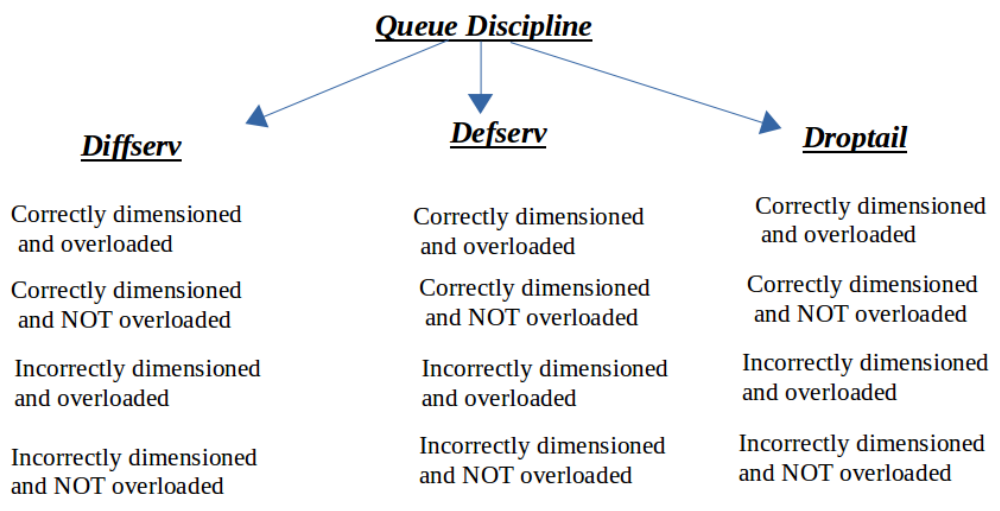

Queue Discipline
Diffserv
Defserv
Droptail
Correctly dimensioned and overloaded
Correctly dimensioned and NOT overloaded
Incorrectly dimensioned and overloaded
Incorrectly dimensioned and NOT overloaded
Correctly dimensioned and overloaded
Correctly dimensioned and NOT overloaded
Incorrectly dimensioned and overloaded
Incorrectly dimensioned and NOT overloaded
Correctly dimensioned and overloaded
Correctly dimensioned and NOT overloaded
Incorrectly dimensioned and overloaded
Incorrectly dimensioned and NOT overloaded

Figure 5.2: The possible network situations.

quite readily. If the network is correctly dimensioned, in this scenario, all flows will experience near-optimal performance.

A more important scenario is where there is one or more large flows present in one or other of the DiffServ classes. Whenever even one large flow is present, the network will be overloaded and performance experienced by all flows during this period will be affected. For simplicity, and to gain understanding, the most important scenarios to explore are where there is one large flow, only, in one of the DiffServ classes. We have also explored more complex scenarios, where there are combinations of more than one large flow, either all in the same DiffServ class, or in a range of classes. However, the behaviour observed in these simulations turns out to be similar to that in the simpler ones.

Because tcp and udp are designed to make the most efficient use of whatever network resources are available, scenarios in which a link is overloaded will occur regularly in the Internet. There may be locations where overload almost never occurs, because all the traffic sharing that link is constrained by the preceding parts of the network through which it passes to get to the link. However, full (and therefore sometimes overloaded) use of transmission resources is a feature of tcp/ip networks that will remain prevalent, especially in access networks, for the forseeable future. The network of Figure 5.1 is designed to model such scenarios.

5.6 Performance analysis of DefServ

Figure 5.3 describes the queue discipline of the mice–and–elephants (`MAE`) algorithm. The highest priority queue is always treated first and only when it is empty is control transferred to lower priority queues. After each packet is treated, in any queue, control is transferred back to the highest priority queue to check if there are any new packets to be treated. In other words, whenever there is a job in a higher priority queue, this job will be done before moving to a lower priority queue.

This queuing mechanism is available in the Linux kernel and in release 3.26 of ns-3, where it is known as `pFifoFast`. The main difference between `MAE` and `pFifoFast` is that in `MAE` the priority assigned to packets is on the basis of the length of the flow in which the packet occurs. Also, in the version of `pFifoFast` in ns-3 there are only 3 queues. It is straightforward to change the number of queues and to allow a range of behaviour to be explored, this has been increased to 8 queues in `MAE`, although we shall see later that this is not actually necessary.

When Defserv uses `MAE`, the traffic will not be classified, in the same way as in DiffServ, and there is no pre-assigned priority for any kind of traffic. Treatment will be only according to *flow size*. The priorities stored in the packets in simulations which use `MAE` will be ignored. The only reason traffic in the different classes appears to experience different performance, in the simulations, is due to random variation from one simulation to the next.

We investigate the relative performance experienced by flows of different sizes. In this chapter, the flow size, in the simulated network, will be identified by packet tagging. These tags are not stored in the IP header, but are associated with each packet in the simulation, only. Ns-3 provides this tagging mechanism in a very general form for the collection of statistics (e.g. end-to-end latency), which would be difficult to measure otherwise.

The flow-size tags used were introduced, in the ns-3 software, specifically for these simulations. In a deployment in the Internet, tagging will not be used. To investigate how a real system can achieve the same results without tags, experiments have been conducted in which the flow-size tags are only *retrospective*, i.e. only the flow size *up to this point* is used in the queue discipline. Also, the number of flow-size thresholds has been reduced and only very large thresholds used. In this way we are able to show that the queue discipline needs only the information about flows which are very large which makes it much easier to carry out the necessary flow size measurements. Software which efficiently measures flow-sizes from traffic traces, or from a sampled live traffic trace, has been already developed (**?**).

To understand the performance of the Defserv architecture, we compare simulations where Defserv is in use with ones where Droptail is the queue discipline and we also compare simulations where Defserv is in use with ones where `pFifoFast` is used, to model the performance of Diffserv.

5.6.1 Defserv vs. Droptail

The simulations described in this chapter can be viewed (and repeated) by visiting the URLs shown in Table 5.1. Note: each time the simulations are run the results will be different. The simulations shown here have all been repeated many times and

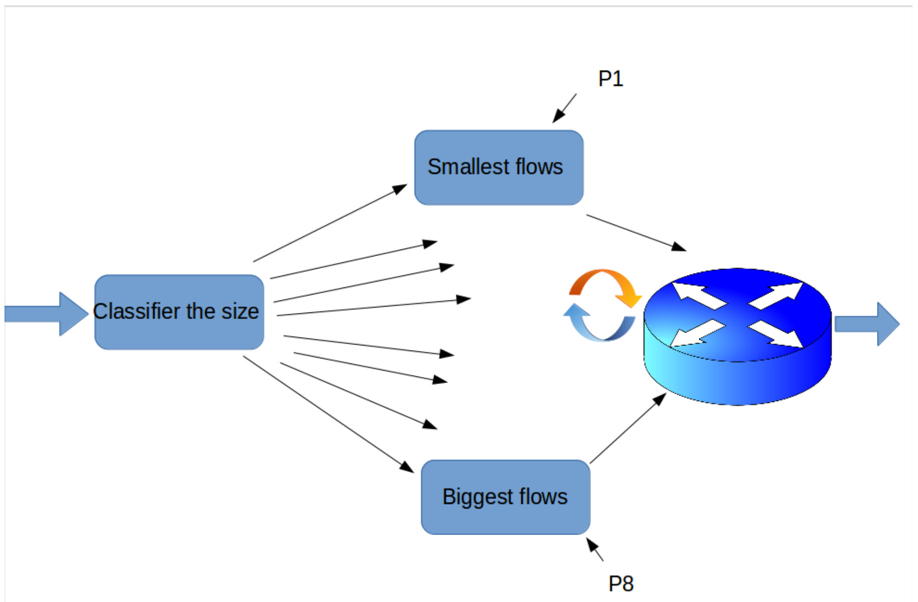


Figure 5.3: The queue discipline of mice–and–elephants (`MAE`) algorithm.

the results shown here show typical outcomes.

In this section we compare Defserv to the traditional Droptail queue discipline. Figures 5.4 and 5.5 show the effective throughput for each flow during a simulation, on the y axis, vs the size of each flow, on the x axis. Each figure shows the results from two different simulations: one in which the queue discipline at the congested link was Droptail, and one in which the queue discipline was MAE. The two simulations were identical in all respects, except the queue discipline. In particular, both were started with the same seed for the random number generator. The seed was randomly chosen, but the same random choice used for any simulations whose results were compared in the same graph. In these figures and Figures 5.8 and 5.7 as well, every point (cross or square) is an individual flow and these flows have the Pareto distributed lengths. Note that time is not relevant in these graphs: the results for each flow have been gathered and sorted by flowsize in order that they can be plotted meaningfully. Since the experience of each flow varies randomly, to make the trend more clear, a mean-square regression has also been carried out, fitting a linear curve

Table 5.1: Simulation experiments

Name of experiment	URL of the experiment	The Figure number
Effective throughput of Droptail vs. MAE queue discipline when the network is not overloaded.	`https://netml.usq.edu.au/netml4_63/index.jsp?netname=mae_Droptail_traffic&location=Users`	5.4
Effective throughput of Droptail vs. MAE queue discipline when the network is overloaded with long–burst stream.	`https://netml.usq.edu.au/netml4_63/index.jsp?errorMsg=In+authenticate.jsp%2c+redirecting+to+index.jsp%3a+you+should+now+be+logged+in.&location=Users`	5.5
The traffic of Defserv vs. Droptail simulations.	`https://netml.usq.edu.au/netml4_63/index.jsp?netname=mae_Droptail_traffic&location=Users`	5.6
Defserv vs. Diffserv with not overloaded network.	`http://netml.usq.edu.au/netml4_63/index.jsp?netname=mae_pfifo_nolong1&location=Demo&userid=albdair`	5.7
Defserv vs. Diffserv with overloaded network.	`http://netml.usq.edu.au/netml4_63/index.jsp?netname=mae_pfifo_long1&location=Demo&userid=albdair`	5.8

to each set of flows, to show the overall trend. These lines can therefore be used to compare the average performance experienced by flows when one queue discipline is used to the average performance experienced when the other queue discipline is used.

Figure 5.4 shows that the mice and elephants queue discipline can sometimes make no difference– when the network is not congested. However, when the network is overloaded, Figure 5.5, the mice and elephants queue discipline can have a dramatic benefit for the performance of the all flows, by comparison with the Droptail queue discipline.

In Figure 5.5 the effective throughput when Defserv is in use, as represented in red colour, is greater by approximately 7 times, on average, than the throughput of Droptail queue discipline which is represented in blue colour. This significant benefit is obtained by the vast majority, about 90%, of small flows. Also, the variance of `MAE` discipline case, Figure 5.5, is much smaller than the variance of Droptail.

Figure 5.6 shows the traffic of the two simulations together on the same graph so that we can check that it is fairly similar. Although the simulations use random numbers starting with the same seed value, because the events which occur in them are not identical, it is not possible to ensure that the traffic is identical. Fortunately the traffic is similar enough that this discrepancy of the traffic is unlikely to affect the observed performance significantly.

5.6.2 Defserv vs. Diffserv

Differentiated services (DiffServ) is an architecture which aims to provide better Quality of Service (QoS) and to allow certain classes of traffic to meet pre-specified service constraints. Within the DiffServ architecture, traffic is classified at entry to

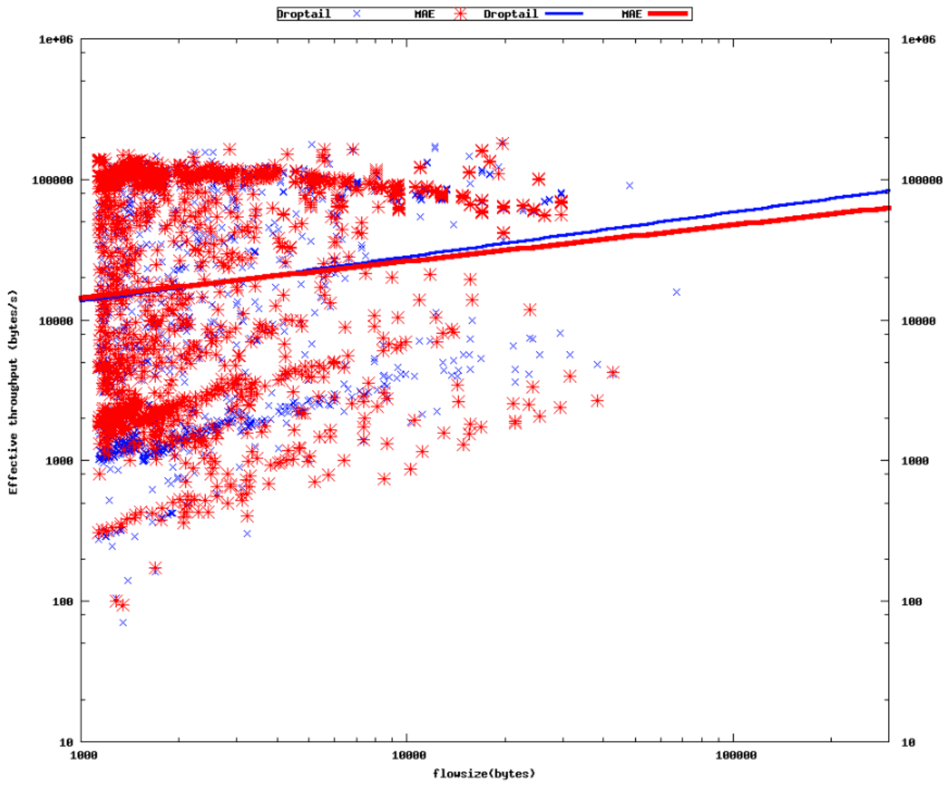


Figure 5.4: Effective throughput of Droptail vs. mice–and–elephants queue discipline when the network is not overloaded. (Simulation: DroptailVsDefservWithNoLong)

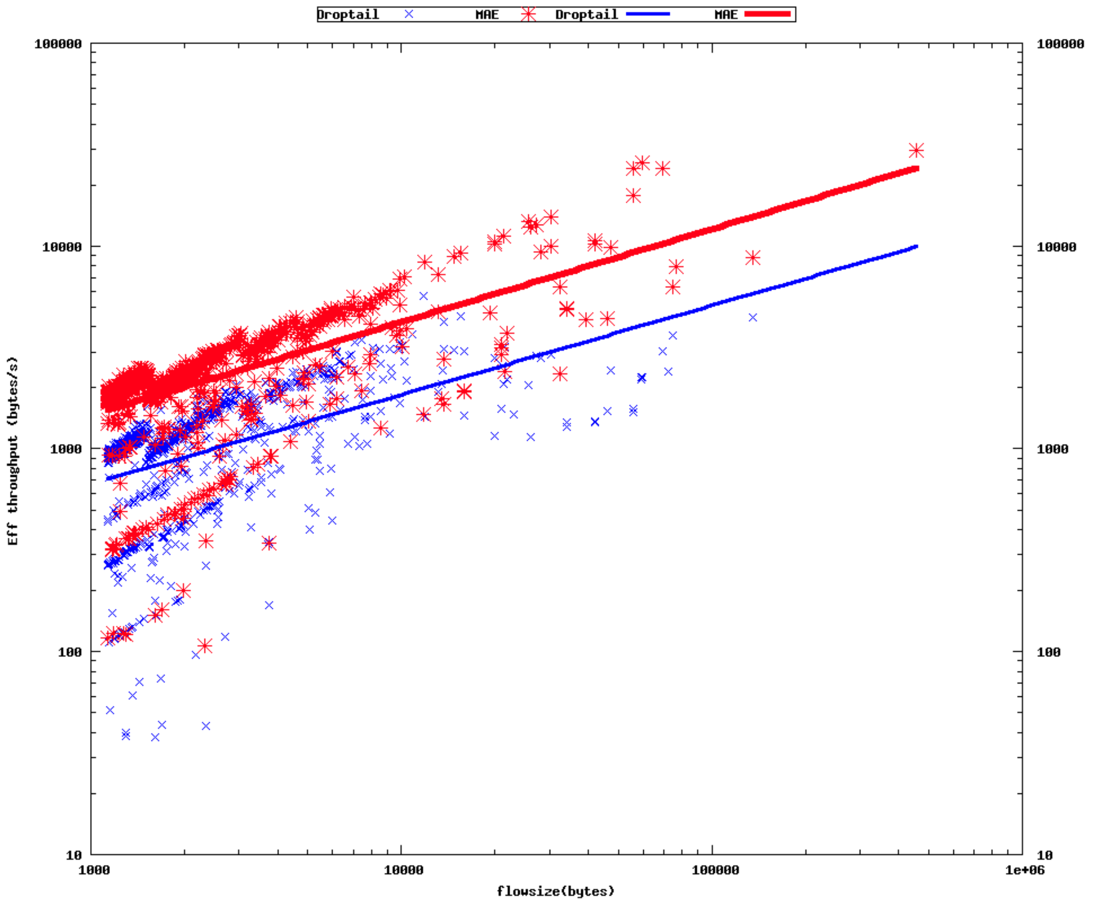


Figure 5.5: Effective throughput of Droptail vs. mice–and–elephants queue discipline when the network is overloaded.

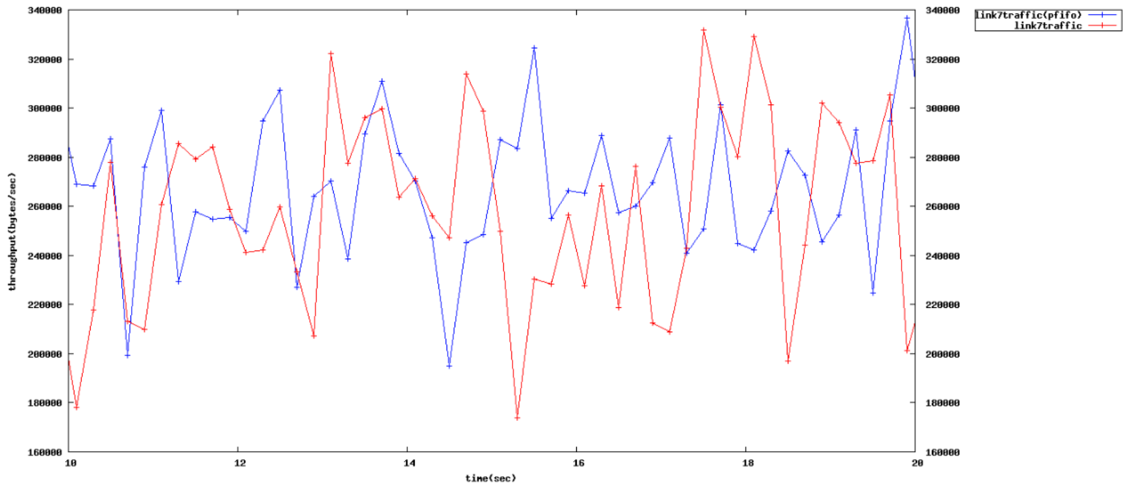


Figure 5.6: The traffic of Defserv vs. Droptail simulations.

the network and then may be reclassified, within the network if the aggregate traffic in this class is more than allocated for at certain points in the network. In each packet six DSCP bits in the IPv4 TOS octet, or in the IPv6 Traffic Class octet, are used to contain the DSCP value. At each hop, packets receive special forwarding treatment, which is called per-hop behavior, according to the value of their DSCP at each network node.

The range of possible per-hop behaviours that can be implemented and used is unlimited. Thus DiffServ cannot be treated as a feature which can be simply turned on or off. However, a key design limitation is that DiffServ tags are assigned at entry and are not dynamically set depending on traffic conditions (except for re-tagging packets which are out-of spec). This limits what can be achieved by DiffServ.

DiffServ can include the combined effect of priority (i.e. packets of one DSCP value are served ahead of packets with a different DSCP value), token bucket policing (so that when traffic with a certain DSCP value exceeds a certain load its DSCP value is changed), and selective early dropping (like RED), where packets are dropped if their position in a buffer falls above a certain threshold.

The `pFifoFast` queue discipline used in the simulations in this chapter only implements the priority aspect of DiffServ. Figures 5.7 illustrates the effective throughput of both `MAE` queue discipline and `pFifoFast` queue discipline for flows as a function of flow size when the network is not overloaded. In this case, most flows experienced good performance. *Notice this is similar to the Defserv vs Droptail scenario* in Figure 5.4.

Figure 5.8 shows the case when the network is overloaded and either `MAE` is used or `pFifoFast`. In both cases, the traffic falls into three classes. When MAE is in use, these classes are ignored, but when pFifoFast is used, Class 0 traffic is given priority over Class 1, which is, in turn, given priority over Class 2. The effective throughput

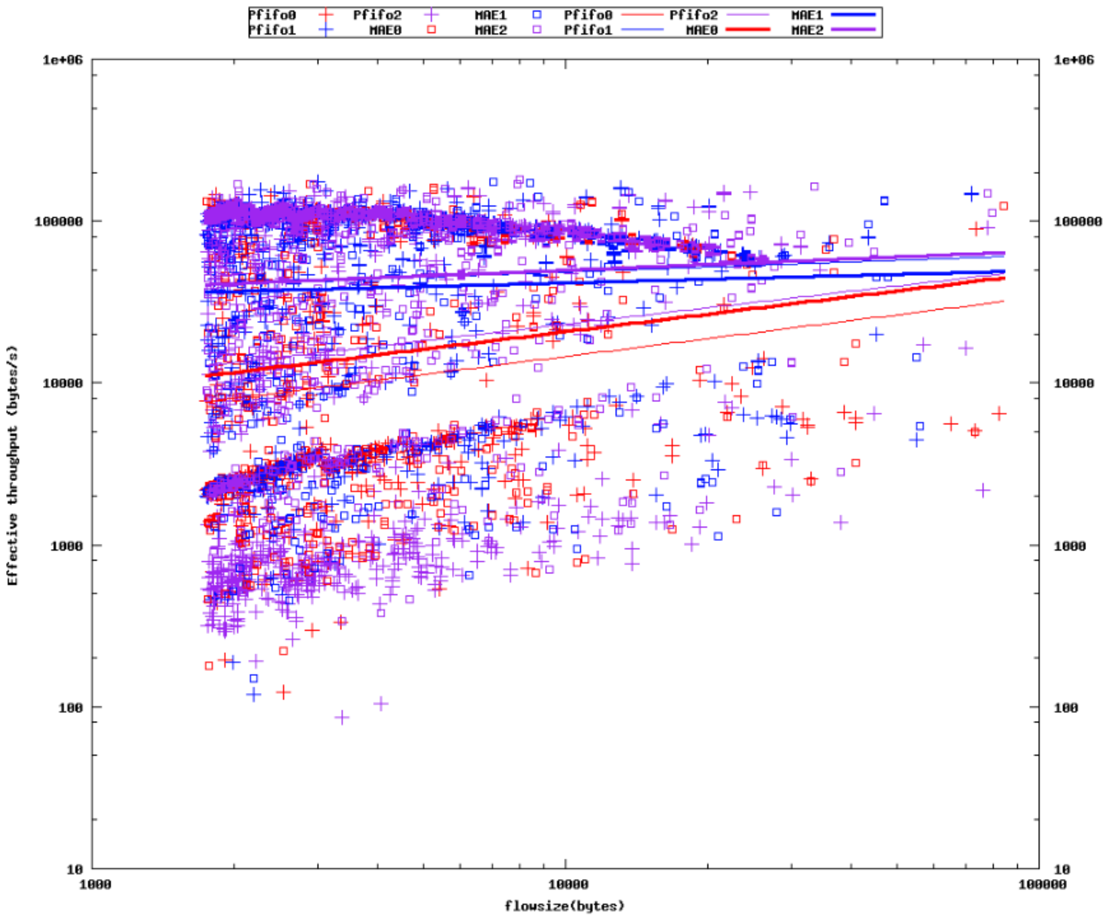


Figure 5.7: Defserv vs. Diffserv with not overloaded network.

of the three high priority classes of traffic is similar in the two queue disciplines, but while the lowest class traffic also receives similar performance under MAE, under pFifoFast, it experiences much worse performance.

Thus, under overload, although DiffServ is unable to deliver better performance for the higher priority traffic classes, than MAE, it is able to deliver worse performance for the lowest priority class.

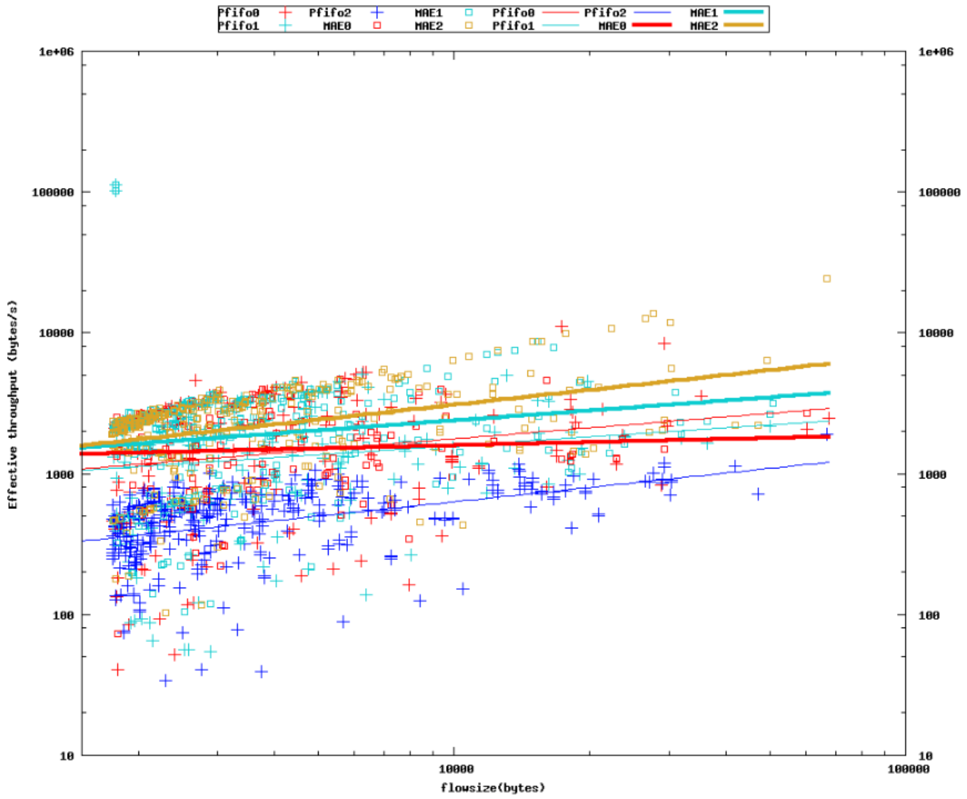


Figure 5.8: Defserv vs. Diffserv with overloaded network.

5.7 Implications

The observations of Chapter 4 show that periodic occurrences of congestion caused by large flows is a phenomenon which can be expected to occur repeatedly simply because of the normal statistics of traffic. This means that periodic overloads of varying severity will certainly occur because of the nature of traffic. These overloads are not due to bad design, or bad traffic, they are just in the nature of real traffic. These overloads will not be confined to any particular class of traffic. Consequently, we should take in the account the way in which the queue disciplines in the routers should be managed, i.e. a strategy which monitors traffic, identifies large flows (or eigenflows), in a statistically sound way, and manages their impact, is essential, to ensure stable performance, and to lower the impact of unusual traffic fluctuations. Thus, a strategy of managing traffic based on statistical observation is needed to produce better Quality of Services from social network behaviour understanding.

5.8 Summary of the chapter

The Defserv architecture accepts and works with the principal that for a good proportion of the time, the current situation, in regard to the traffic which is present in a network, will have a great influence on the performance experienced by users. As a consequence, it is essential that the traffic management strategies dynamically adapt rather than enforcing the same rules all the time. In this chapter a queue discipline called Mice and Elephants (`MAE`) has been defined, and a QoS architecture called *Defserv*, based on this queue discipline, has been tested using simulations using ns-3. The Defserv architecture has been shown to have significant benefits.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

A Pareto principle was introduced in Chapters 3 and 4 to apply to every aspect of the traffic which has been considered. The range of statistical characteristics considered includes most, or at least many, of the possible ways in which social behaviour can be inferred from traffic.

Most users contribute very few bytes, and most bytes are from a very small proportion of users. This applies to sources, destinations, and O-D pairs in Chapter 3 and applies to the eigenflow analysis of traffic matrices in Chapter 4. The degree to which this is the case can be measured. It is the slope in the curves shown in Figures 3.2–3.4. These parameters are characteristic of the underlying social network (without in any way referencing any single individual or flow) which we can measure, and investigate.

To convey the quality of power law distributions we have frequently emphasised the extreme values and this may give a false impression. It is not only the fact that a high proportion of bytes are accounted for by a small number of flows, while another (different) high proportion of flows account for a very small proportion of bytes, which is revealed in Figures 3.6–3.8. The power-law character applies across the full range of flow sizes and this is likely to be important in order to be able to apply it effectively.

The statistical analyses presented in Chapter 3, show that there are important characteristics of the underlying social network which can be identified from traffic data. Also, we see in Figure 3.11, a slightly different characterisation of social behaviour is revealed. These features measure the degree to which all communication is mediated through a limited number of gatekeepers.

In Chapter 4 we discovered that the traffic has some meaningful features which, potentially, play a vital role in changing the way we manage and control the networks effectively. The most interesting phenomenon which has been discussed and analysed in this chapter is the concept of an eigenflow. Singular value decomposition of several different types of traffic matrices was used to identify eigenflows. We discovered that there are a small number of very large eigenflows dominate the network at any one time.

Three types of matrices are used in Chapter 4; mean, variance and covariance to analyse four groups of data sets obtained from CAIDA. All the experiments of this chapter illustrate the Pareto principle, which is that a small proportion of eigenflows explains high proportion of the traffic. For example, there is 2.5 % from all eigenflows of covariance traffic matrix, explains about 75 % of the traffic, in bytes^2 , of covariance matrix. This is true for all data-sets.

It has been suggested (?) that these eigenflows may reveal important features of network traffic. The significance of the eigenflows, in Chapter 4, has been tested by comparison with traces from simulations in which traffic flows are all independent.

Under this hypothesis we found that the results are significantly different from the behaviour observed from CAIDA traces. Also, the Pareto shape parameters (γ) have been estimated for all the curves from the CAIDA data sets and the simulation results to show how *heavily* the Pareto principle applies in each case.

Estimation of the underlying social network within the Internet has revealed critical features which have a dramatic effect on its behaviour and in particular the performance experienced by users. Traffic has a very pronounced Pareto characteristic, which means that quite often there will be a relatively small number of large flows which have a highly significant impact on other users.

In chapter 5 a queue discipline called Mice and Elephants (`MAE`) has been defined, and a QoS architecture called *Defserv*, based on this queue discipline. The Defserv architecture has been tested using simulations using ns-3. The benefits of DefServ relative to Doptail and DiffServ are shown in Figures 5.4–5.8. Hundreds of simulations have been carried out in the preparation of this chapter. The four simulations presented here demonstrate that: (a) when there is no congestion, performance is much the same under DiffServ, DefServ, and Droptail; and, when there is unusally high traffic, which is an irregular, but normal occurrence, Defserv is able to maintain relatively good performance for all traffic classes, and all flow sizes, whereas Droptail provides significantly worse performance for short flows. Other simulations show that under overload, when DiffServ is used, the lowest priority class experiences significantly worse performance, for all flow sizes. In all simulations, small flows experienced better performance when Defserv was used relative to when it was not. The experimental results illustrate significant benefits from the use of Defserv in the performance experienced by small flows, when a link is overloaded.

In Chapters 3 and 4, it was shown that traffic has a strongly Pareto character, and that this Pareto quality is exhibited in a multi-facted way. The activity of sources, and of destinations, and of O-D pairs all independently show this Pareto quality as illustrated in Figures 3.2 – 3.4. This implies by itself that the approach to traffic management described in Chapter 5 is particularly attractive. In addition, it was shown that when a spectral analysis of traffic is carried out features of the current traffic at any time exhibit eigensources, eigendestinations, and eigenflows which also have a Pareto quality in the sense that a very high proportion of the volume and variation of traffic can be explained by a relatively small number of these sources, destinations, and flows as shown in Figures 4.7, 4.10 and 4.13.

An advantage of these features is that, as statistical characteristics, they can't be disguised or hidden from view (without actually suppressing the communication activity itself). Thus, if the MAE type of traffic management makes use of the spectral analysis of traffic to guides its actions, it will be more robust, and will adapt to the true needs of users more successfully.

The Footbal World Cup is an event, in the Internet, of global importance. The chief executive of Optus Allen Lew said "on Wednesday the company 'deeply regretted' service had not been up to the standard expected and would allow SBS to broadcast all games until the end of the group stage of the tournament" (**?**). According to the Pareto model of eigensources, the World Cup is not a one–off event that we need to address by unique methods, but one among a whole series of large events that we need to plan for, both the ones, like the World Cup, that can be predicted, and those that cannot be predicted in advance. Such events are already causing disruption in our Australian Internet and will continue to do so until the Pareto character of whole-of-network traffic is understood.

6.2 Future Work

The Pareto principle as it applies to a whole network implies that we should expect networks to experience a succession of network-wide events of larger and larger sizes. This is a striking observation with important implications, which need to be investigated.

In Chapter 3, we found that the frequency statistics of origin, destination and O-D Pairs of the underlying social network from the Internet trace datasets obtained from the (CAIDA) exhibits the Pareto principle. Also, this pattern has been observed, in Chapter 4, in the behaviour of eigensources, eigendistenations and eigenflows. Hence, in the future we need to answer these questions; will the same behaviour be observed in different places, and at different times? can we use these characteristics to better understand how to serve the needs of the communities using our networks?

A very strong Pareto behaviour signal has been detected in a great variety of ways. The Pareto model implies, therefore, that *extremely large* traffic bursts are likely to occur regularly. Indeed, recent user experience of network outages and service disruptions suggests that such events are already with us. Network operators should, therefore, incorporate traffic control strategies to manage these extreme traffic bursts. Research into efficient, robust strategies, based on the traffic analysis methods and algorithms developed in this project, needs to be undertaken.

Because of our knowledge of the underlying social network, it is clear that periodic overloads are not a sign of bad design, but are a normal feature of Internet traffic. Although we have spent considerable time *comparing* Defserv to DiffServ, it is also possible to use both at once, or to view Defserv as new type of PHB which can be deployed in routers. While DiffServ introduces the concept of differentiated treatment of traffic based on the class of the target traffic, DefServ introduces the idea of differentiated treatment based on the current mix of dominant traffic streams, and the volume and nature of these traffics. Investigation of the benefits of such an integrated approach will be considered in future research.

Appendix A

Antraff

In this appendix we provide the central class in the *Antraff* software called `ipbin`. The parameter of this template class is the object used to contain an address of the objects under analysis. Thus, when statistics concerning sources are being analysed, the objects under analysis are sources, and the address of a source is an IP address, so the ipbin class used is `ipbin<u_int32_t>`. Similarly, when statistics concerning destinations are being compiled and analysed, the ipbin class used in also `ipbin<u_int32_t>`. When statistics concerning O-D pairs are being compiled and analysed, the ipbin class used is `ipbin<pair<u_int32_t,u_int32_t>>>`. In order to carry out a similar analysis which differentiates different O-D pairs on the basis of their DSCP code point value, the ipbin class used will need to be `ipbin<triple<u_int32_t,u_int32_t,u_int8_t>>>`.

Listing 1: Header file for the class ipbin.

Appendix B

Experimental Procedure

The process of conducting the experiments of Chapter 3 and Chapter 4 is done by using *Antraff* software (**?**). In this appendix we explain, as an example, the process of one of those experiments. The experiments of Subsection 3.5.2 are done, like all other experiments in this work, in the High Performance Computing (HPC) facility of the University of Southern Queensland.

Two groups of data have been used in these experiments; the traffic traces obtained from CAIDA and a data file which was generated by simulation. The first step is to make a directory where the experiment will be conducted.

Listing 2 shows the commands for carrying out an experiment, and the resulting output. The command `rmake` is the same as the standard Unix `make` command except that it refers to a target in a makefile in a different directory from where the command is issued, namely the Antraff makefile (See Appendix C). The `rmake` command in Listing 2 seeks to carry out all steps to make the target (*allfrequencies.pdf*) according to the steps recorded, in the Antraff *makefile*. The next step, after creating a directory and switching into it, is that we set some environmental variables which define the free parameters of this experiment. These settings are stored in a file

.

Listing 2: An example experiment

(called al), which documents the settings for this particular experiment and makes it easy to re-run the experiment.

The third and last step is to use `rmake` to undertake all the processes required to generate a specific target. The script `rmake`, which is documented in (**?**), makes a target, as specified in the standard makefile (see Appendix C), but does so in the directory where this experiment is being conducted. The waiting time depends on the nature of the experiment; the SVD calculation, for example, takes $4 - 6$ hours to produce the required plots.

Appendix C

Makefile

This Appendix provides a listing of the Antraff software makefile This file includes instruction for how to make many Antraff targets. All the listed targets in makefile can be made by entering the command `make target` in the directory where the Antraff source code is located. Using make in this way only would force all work to have to occur in one directory, which would make experiments difficult to manage. For this reason, a script, `rmake`, has been developed which allows targets in the Antraff makefile to be made in other directories also.

Listing 3: Makefile for the antraff package.