Delay estimation in computer networks

Nicholas Alexander Johnson

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
December 2009

Abstract

Computer networks are becoming increasingly large and complex; more so with the recent penetration of the internet into all walks of life. It is essential to be able to monitor and to analyse networks in a timely and efficient manner; to extract important metrics and measurements and to do so in a way which does not unduly disturb or affect the performance of the network under test. Network tomography is one possible method to accomplish these aims. Drawing upon the principles of statistical inference, it is often possible to determine the statistical properties of either the links or the paths of the network, whichever is desired, by measuring at the most convenient points thus reducing the effort required. In particular, bottleneck-link detection methods in which estimates of the delay distributions on network links are inferred from measurements made at end-points on network paths, are examined as a means to determine which links of the network are experiencing the highest delay.

Initially two published methods, one based upon a single Gaussian distribution and the other based upon the method-of-moments, are examined by comparing their performance using three metrics: robustness to scaling, bottleneck detection accuracy and computational complexity. Whilst there are many published algorithms, there is little literature in which said algorithms are objectively compared. In this thesis, two network topologies are considered, each with three configurations in order to determine performance in six scenarios. Two new estimation methods are then introduced, both based on Gaussian mixture models which are believed to offer an advantage over existing methods in certain scenarios. Computationally, a mixture model algorithm is much more complex than a simple parametric algorithm but the flexibility in modelling an arbitrary distribution is vastly increased. Better model accuracy potentially leads to more accurate estimation and detection of the bottleneck.

The concept of increasing flexibility is again considered by using a Pearson type-1 distribution as an alternative to the single Gaussian distribution. This increases the flexibility but with a reduced complexity when compared with mixture model approaches which necessitate the use of iterative approximation methods. A hybrid approach is also considered where the method-of-moments is combined with the Pearson type-1 method in order to circumvent problems with the output stage of the former. This algorithm has a higher variance than the method-of-moments but the output stage is more convenient for manipulation. Also considered is a new approach to detection algorithms which is not dependant on any a-priori parameter selection and makes use of the Kullback-Leibler divergence. The results show that it accomplishes its aim but is not robust enough to replace the current algorithms.

Delay estimation is then cast in a different role, as an integral part of an algorithm to correlate input and output streams in an anonymising network such as the onion router (TOR). TOR is used by users in an attempt to conceal network traffic from observation. Breaking the encryption protocols used is not possible without significant effort but by correlating the un-encrypted input and output streams from the TOR network, it is possible to provide a degree of certainty about the ownership of traffic streams. The delay model is essential as the network is treated as providing a pseudo-random delay to each packet; having an accurate model allows the algorithm to better correlate the streams.

Declaration of originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the School of Engineering at The University of Edinburgh.

Nicholas Alexander Johnson

Acknowledgements

There are many people who have helped me pursue and finish my PhD. Without them, I would surely have never made it this far. The following is a list of those who come to mind but there are doubtless others whom I have forgotten. To them, I am grateful for your friendship and support and am sorry I haven't listed you by name.

Firstly, I must thank my parents and family for their unwavering support. Also my supervisors, Steve McLaughlin and John Thompson who have offered support, advice and encouragement throughout my work. Thanks must also go to Francisco Garcia at Agilent Technologies for helping me to get started and secure funding.

I would like to thank my colleagues in the Signals and Systems group who had to put up with my moans, make me tea and go to lunch with me for the past four years. Special thanks are due to Sharad Nagappa, Mostafa Afgani, Sinan Sinanović and Aby Iyer who have bourne the brunt of this load.

I also thank all my colleagues, past and present, in Pollock Halls who had the unenviable task of having to listen to me moan at dinner after some frustrating days in the lab, cover for me during trips and take over my wardennial work when I couldn't find time. Special thanks are due to Charles, Sam, Francesca, Louise and Raphie in this regard.

I am grateful to my friends Ivona Vicković for making sure I kept going when I would otherwise have given up, and Leonard Humphries and his family for allowing me to escape from the world of the university during each holiday for the past ten years.

Finally, I must acknowledge support from Agilent Technologies in respect of my Agilent Technologies Fellowship Award.

Contents

List of figures

List of tables

Acronyms and abbreviations

ABW	available bandwidth
ATM	asynchronous transfer mode
BGP	border gateway protocol
CBR	constant bit rate
CBR	constant bit-rate
CCDF	complimentary cumulative distribution function
CDF	cumulative distribution function
CGF	cumulant generating function
CS	compressed sensing
DHCP	dynamic host control protocol
DNS	domain name system
EM	expectation maximization
erfc	complementary error-function
FD	f-divergence
FIFO	first-in first-out
FTP	file transfer protocol
GA	single Gaussian distribution
GCA	Gram-Charlier A series
GMM	Gaussian mixture model
GSM	global system for mobile

HSCSD	high speed circuit switched data
IP	internet protocol
ISDN	integrated services digital network
IS-IS	intermediate system to intermediate system
KLD	Kullback-Leibler divergence
KS	Kolmogorov-Smirnov
LL	link layer
LS	least squares
MAC	medium access control layer
Mbps	mega bits per second
MGF	moment generating function
MLE	maximum likelihood estimate
ML	maximum likelihood
MOM	method of moments
ns2	network simulator version 2
OD	origin-destination
OSI	open systems interconnection reference model
OSPF	open shortest path first
PASTA	Poisson arrivals see time average
PDF	probability density function
PHY	physical layer
PMF	probability mass function
PMF	probability mass function

P-MOM	Pearson method-of-moments
POTS	plain old telephone service
PRS	Pearson type-1 distribution
QOS	quality of service
RM	routing matrix
RP	routing protocol
RT	raw transform
RTT	round-trip time
S3	Scalable Self-Organizing Simulation
SD	source-destination
SMC	sequential monte-carlo
SOGA	Sum-of-Gaussians A
SOGB	Sum-of-Gaussians B
TCP	transmission control protocol
Tor	the onion router
TTL	time to live
UDP	user datagram protocol
VC	virtual circuit
VOIP	voice over internet protocol

Nomenclature

P	The number of paths in a network
L	The number of link in a network
Y_{ik}	The k^{th} delay on path i
X_j	The delay distribution on link j
$\widehat{\mu_{X_j}}$	The estimated mean of the delay on link j
$\widehat{\sigma^2_{X_j}}$	The estimated variance of the delay on link j
$\widehat{\mu_{3_{X_j}}}$	The estimated skewness of the delay on link j
$\widehat{\mu_{4_{X_j}}}$	The estimated kurtosis of the delay on link j
Y_i	The delay distribution on path i
H	The routing matrix
H^{-1}	The inverse routing matrix
h_{ij}	The element of the inverse routing matrix at row i and j
κ_n	The n^{th} cumulant
μ'_n	The n^{th} central moment
$\widehat{K_{X_j}}$	The estimated cumulant generating function for link j
$\widehat{M_{Y_i}(t)}$	The estimated moment generating function for path i at time t
P_j	The probability of link j being the bottleneck link
J	The number of elements in a Gaussian mixture model
δ	The delay threshold
γ_1	The skewness
$\widehat{\gamma_1}$	The estimated skewness
γ_2	The kurtosis
$\widehat{\gamma_2}$	The estimated kurtosis
N	The number of samples of data
B_t	The block size used in RT
C_X	The estimated distribution for X
H_X	The hypothesis that distribution X contains the input stream
$d(x)$	The delay function
B	The number of blocks
β_1, β_2	The defining parameters for a Pearson distribution

Chapter 1
Introduction

This thesis is concerned with the application of signal processing algorithms to the problem of estimating delay in computer networks. Network tomography [1] describes the process of inferring some characteristics [2] of the internal links of a packet-switched network using measurements of the path level characteristics and knowledge of the topology and the routing of the network. Algorithms and techniques are in demand because they offer a means to monitor large networks without the necessity of measuring every link; thus, they can offer a saving of time and resources. There are many published algorithms [3] [4] [5] [6] whose authors claim good performance but as yet no comparisons between algorithms using identical data and scenarios have been made; thus an objective choice of algorithm based on performance is hard.

Methods of network tomography can be broken down into two parts: an estimation stage and a detection stage. In general, research effort is focused on the estimation stage because this offers greater scope for the application of signal processing algorithms. In addition, multiple detection algorithms can be coupled with the output from a single estimator stage to focus on detecting different anomalies; hence accurate estimation is highly important. Algorithms for both detection and estimation are examined in this thesis and a common detection method suitable for all estimation algorithms is sought.

The difference between parametric methods and non-parametric methods can be described thus: parametric methods are generally less computationally complex but are less accurate, non-parametric methods are potentially more accurate but are also more computationally complex. In this thesis both classes are examined, initially using the method of moments (MOM) [3] as an example of non-parametric methods and a single Gaussian distribution (GA) as an example of parametric methods in a direct comparison of performance using the same simulated data. Both classes of algorithm are developed by extending the parametric distributions to include Gaussian mixture models (GMMs) and the Pearson type-1 distribution (PRS) as well as a hybrid method, the Pearson method-of-moments (P-MOM). That is the core

aim of this work - to compare estimation and detection algorithms in a controlled environment where their performance can be objectively assessed.

One problem encountered when assessing such performance is in determining which metrics to use to compare the methods under examination. Sadly, the literature offers no common approach - this leads to the development of the metrics used in this thesis. Based on the knowledge that algorithms are designed to operate in real-world scenarios of an *a-priori* unknown scale, bottleneck link detection accuracy, computational complexity and robustness to scaling are considered best suited to evaluating performance.

Whilst offering a test-ground for the algorithms, network tomography is not the only application area for delay estimation. Algorithms can be applied to real-world data from a variety of sources and used to estimate functions which can form part of a larger algorithms. Tracking algorithms for anonymising networks is one example where having an estimate of the transfer function of the network is immensely valuable and forms a part of the larger algorithm.

1.1 Problem description

The problem of network tomography is related to the size and scope of networks used. It is often impossible for reasons of cost, access or ownership to measure each link on a network such as the Internet. Whilst it may be possible to do so on a privately owned network it may be impracticable to do so if access to the endpoints of any links is difficult. Network tomography therefore provides a solution to both these problems, measurements can be made where possible and the desired information, i.e. the information about the link or links in question, can be inferred from the available data. The cost associated with this is a necessity to know the topology of the network under test. For a fixed, wired network, it is not unreasonable to assume this is known or can be discovered.

In this thesis, work centres on inferring the delay distribution on network links using measurements made on network paths. The scenarios used are small such that the mathematics is tractable and algorithms examinable but it is possible to foresee how the results would apply to large networks or the Internet. Figure 1.1 shows an example network topology with four links and five paths and where the desire is to estimate the delay distribution for each of the links where measurements can only be made at the end points of the paths. In this thesis, a path is assumed to be a connected set of links. The relationship between the links and the

paths is shown in the routing table, Table 1.1 where a 1 or a 0 in each position indicates whether or not a link (column) is part of a path (row). This table must be transformed using the pseudo-inverse in order to perform the inference of link delay distributions from path delay distributions.

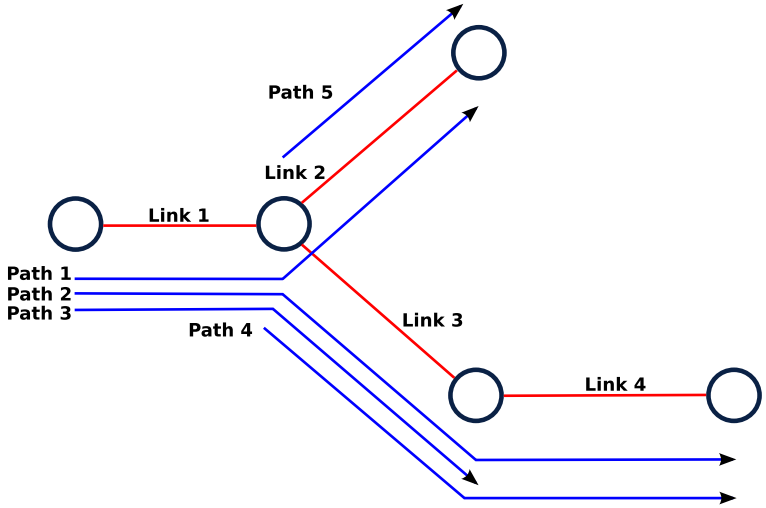


Figure 1.1: *The topology of the 5 node network with 4 links (shown in red) and 5 probe paths (shown in blue). This is based on the topology used in [1] [3].*

		Link			
		1	2	3	4
Path	1	1	1	0	0
	2	1	0	1	1
	3	1	0	1	0
	4	0	0	1	1
	5	0	1	0	0

Table 1.1: *Routing table for the network topology shown in Figure 1.1. The columns depict the link and the rows depict the paths. A 1 or a 0 indicates the inclusion or non-inclusion of a link in a path.*

1.2 Contributions

Four contributions are made in this thesis. First, a comparison between two published methods, the GA and the MOM is given. Whilst the performance of methods is given in the the papers introducing them, no direct comparisons between these two particular methods have been

published. Second, alternative methods for modelling path and link delay distributions are proposed and examined, this includes using GMMs and the PRS. Third, a new detection method which seeks to remove the dependence of the parametric algorithms on the delay threshold value is intoduced. Whilst the mathematics of the method is not novel, such a method has not been demonstrated in network tomogrpahy literature. The method is tested using both synthetic data and data from network simulation to assess its viability. Finally, an implementation of an algorithm [7] for tracking users in the onion router (Tor) [8] [9] network is shown along with results gained by applying this algorithm to real network data; something which had not been performed when the algorithm was introduced.

1.3 Thesis layout

The remainder of this chapter describes the structure of the remainder of this thesis which is split into a further seven chapters.

Chapter 2 begins with a discussion about the different types of modern large-scale networks that are used to transmit information. The protocols that are necessary to enable efficient use of the network are considered along with the open systems interconnection reference model (OSI) [10] of packet-switched networks which makes it possible to abstract away from any hardware specific details and allow discussion of networks in general. There is a review of some concepts and ideas such as sampling and routing in networks, the key least squares (LS) algorithm and its application in the context of network tomography and methods of manipulating distribution functions. Also included is brief taxonomy of some of the published methods used in network tomography.

Chapter 3 shows, through example, how the data used in the simulations in later chapters is generated, processed and analysed. The process is comprised of three stages: simulation, data processing and signal processing. Each stage requires a different tool, ns2, AWK and MATLAB respectively. How data is converted between various formats is also examined.

Chapter 4 provides the first technical material, a direct comparison between the performance of the MOM and a GA in six different scenarios. Both algorithms are examined and a subjective analysis of their performance is conducted using the three metrics introduced here. Details of the simulation parameters chosen are discussed in order to define a fair test of both algorithms and to ensure any experimental bias is minimised.

Chapter 5 develops the GA estimation algorithm, expanding it to use a GMM to increase flexibility in the estimation stage. Using mixture models in this context is not novel but the use of GMMs is. The impact that the use of a mixture model has on all metrics is considered, and by comparison to GA, whether or not the cost of the extra computational complexity is outweighed by an increase in performance.

Chapter 6 examines methods of improving network tomography algorithms. Consideration is given to the PRS as a flexible alternative to the Gaussian distribution. Two uses of the PRS are examined: using it as a replacement for the Gaussian in an algorithm similar to GA, and as an output stage which is coupled with a MOM front-end. It is concluded that the latter, hybrid approach offers a good compromise between the computationally intensive MOM and the flexible Pearson. Secondly, the possibility of using the Kullback-Leibler divergence (KLD) [11] [12] as a detection stage is discussed. Using the KLD can alleviate the necessity of *a-priori* definition of the delay threshold, δ.

Chapter 7 departs from the previous work involving network tomography to consider delay estimation as part of an algorithm for tracking users through anonymising networks such as Tor. The algorithm proposed by Danezis [7] is examined and its implementation tested on a real-world network to probe its working and the assumptions made about it (and indeed about how to implement it) and to generate some results. The algorithm is then adapted to prove the concept of a tree-based search to identify the route taken by packets across the whole network.

Chapter 8 concludes the thesis with some final observations, highlighting the main contributions of the thesis and presents a short-list of possible areas for future work.

Chapter 2
Background material

Introduction

The desire to measure and analyse the performance of networks has existed for as long as network technology itself and many people have proposed efficient and robust methods and systems to do so. There are a number of problems which are still to be addressed in this area including the collation of data, timely analysis and efficient measurement strategies. It is this last point which is the motivation for much of the work in this thesis.

This chapter is divided into four section. The first section (2.1) provides a brief introduction to the technology and types of network that are currently prevalent, how they have arisen and some of the terminology associated with them. The next section (2.2) gives an overview of internet protocol (IP) layer measurement including the important *What, Where and When* questions. The third section (2.3) provides a short taxonomy of inference methodology with an examination of current schemes. Ultimately, in section four (2.4) there is a review of some general concepts which are of relevance throughout the thesis and are well known but require setting in the context of this work.

2.1 A description of data networks

There are currently two types of data network that are likely to be deployed. The older type, circuit-switched networks, have largely been replaced by networks of the newer, packet-switched, type. This section first describes a model which describes networks and network operations before examining both network types and ends with a short overview of network routing.

2.1.1 The OSI model of networks

As a result of the complexity of data networks, their connexions and inter-connexions and because the information flowing in the network has to contain data suitable for interacting with different processes and devices at different times, it is necessary to have a model which defines communications between various entities (routers for example). One model is the open systems interconnection reference model (OSI) [10] which is an abstract model for network design which separates functions into one of seven layers (physical, data-link, network, transport, session, presentation and application) to encourage modular design and interoperability. In this model, all network communication between devices takes place between equal layers. So, a network layer at one device can be considered to communicate directly with the network layer at another device. In practice, the network layer sends its data to the next layer down (data-link) which adds some information before sending it down a further layer (physical). At the physical layer the packet of binary digits is transmitted via the physical hardware and received by the hardware at the next device where the reverse occurs (each layer reads the data its counterpart added and decides what to do with the packet) until the packet reaches the target layer at the destination device. Figure 2.1 shows this graphically.

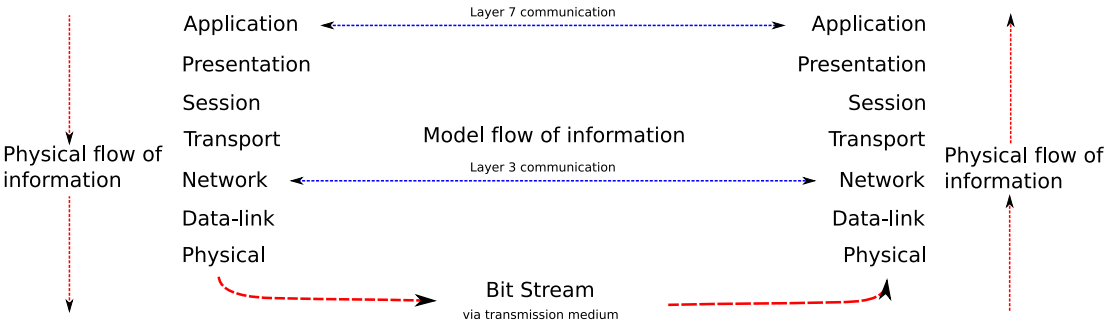


Figure 2.1: *The OSI model showing the physical path data takes between the application layer on the left and the application layer on the right (shown in red) and the abstract view offered by the model of direct communication between the layers (shown in blue).*

2.1.2 Circuit-switched networks

A circuit-switching scheme is perhaps the simplest method to share a link between a number of users subject to the constraint that not all users wish to use it at once. Within a

circuit-switching scheme, a route is established between end users (or callers, if using the telephone nomenclature) using a number of intermediate hops with the distinction that the route established is used for the sole purpose of communication between said users, regardless of how much information they exchange.

The plain old telephone service (POTS) is one example of this, a user connects to their most local exchange (hub) which directs the call to another exchange which may be either: the exchange closest to the call recipient or, an exchange which can forward the call to the exchange closest to the recipient. Even if the line is idle, i.e. the callers are not speaking, the circuit formed is still reserved for their use and no part of the bandwidth can be reused between the initiation (call setup) and cessation (call teardown). There are two necessities for the system to work: one, the network must be organised in such as way as to allow an exchange to be able to route a call towards another exchange it is not directly connected to, i.e. using a routing protocol (RP) and two, there must be enough capacity, i.e. lines, to be able to handle the peak number of calls which must be decided upon in advance (or later added at presumably a higher cost). The advantage is that the quality-of-service is guaranteed, a pseudo-physical[1] link between callers with all of the bandwidth available to them for the call duration and the delay is very likely to be constant. Since this work concentrates on data communication, these points are particularly noteworthy.

Aside from POTS, circuit-switching is used in networks such as integrated services digital network (ISDN) [13] [14] which provides both voice and data services over a single line with a fixed bandwidth and high speed circuit switched data (HSCSD) [15], an upgrade to the data transmission mechanism for GSM mobile services.

2.1.3 Packet-switched networks

The major problem with circuit switched networks is that they pre-allocate bandwidth and that bandwidth cannot be shared or re-used for the duration of the transmission (call). This is useful for voice and situations where a fixed bandwidth must be guaranteed; however, since most data transmission is bursty in nature, it is estimated that up to 90% of the bandwidth is wasted on a circuit [16]. The main concept behind packet-switching is that information

[1] A modern circuit is unlikely to be mechanically connected and will almost certainly include buffers, line-drivers and other associated circuitry. The term pseudo-physical is used to indicate that the circuit functions as if it were physically connected; only in very old systems would the connexion be such.

flowing on a network is divided into small packets of data which can be routed depending on their destination. This offers two advantages: one, the network is distributed in nature and does not require central co-ordination and two, the bandwidth can be dynamically allocated, ie shared, on a link making it between three and one hundred times more efficient in reducing wasted bandwidth than circuit-switching, depending on the type of data being sent. The distributed nature of the network makes for efficient scaling and resiliency to any part of the network being out of service - this was a critical consideration as the earliest networks were for defence departments who were concerned that loss of any hub could seriously affect communications using a circuit-switched network. The efficiency aspect was also important, laying of new lines was costly so any technique which reduced or eliminated the need for new lines was preferable. Because of market economics, it became more efficient around 1969 to employ packet-switching dynamic allocation techniques to increase capacity rather than circuit-switching fixed bandwidth techniques.

The protocols to control a packet-switched network are numerous, each layer requires a specific protocol and some are tied to the transmission medium used. To retain a degree of abstraction from hardware, this thesis concentrates on layer 3 and above. The most commonly used layer three protocol in a modern packet-switched network is the internet protocol (IP) [17] (updated [18]). IP is a connectionless protocol in that it does not require a source to resolve the route to a destination with which it has not previously communicated before beginning transmission. It relies on the addressing scheme to allow it to determine the relevant node to forward packets to. IP is unreliable which means that it does not guarantee packets will arrive in order, error-free, non-duplicated or even at all. Each node uses a best effort approach to deal with the packets it receives leaving resolution of these issues to other layers. Because it does not strictly adhere to the OSI model, IP cannot be used with an arbitrary choice of underlying protocols and can only be paired with the transmission control protocol (TCP) [19] [20] [21].

2.1.3.1 Virtual circuits and the transmission control protocol

There is one disadvantage to the packet-switched networks described above which is that they do not provide a direct link between end points, something inherent in circuit-switched networks. One method to replicate this functionality over packet-switched networks is via the use of a technique called the virtual circuit (VC). There are many specific protocols to accomplish a VC using an unreliable packet-switched network such as one based upon IP,

X.25 [22] or asynchronous transfer mode (ATM) [23]. These techniques are referred to as unreliable because they do not guarantee the delivery of packet data. Most VC protocols provide reliable communication between end points by ensuring that packets arrive error-free and in the correct order using techniques such as automatic re-transmission of erroneous packets (ARQ [24]). VCs add to the communication overhead by requiring signalling between end points (to implement ARQ) and to set-up the path (if the protocol uses constant paths - X.25 and ATM) however the advantage they can offer is a guaranteed quality of service (QOS) - X.25.

The TCP is a protocol designed to operate at the transport layer (using the OSI model) to provide a service between two end points (i.e. a server and a client) which ensures packet delivery is both reliable and ordered. Retaining consistency with the discussion above, TCP can be said to provide a VC to applications therefore making it suitable for use where reliable transmission is necessary such as file transfer, email and web browsing (although the latter two are a specific case of the first).

The TCP establishes a VC but unlike the X.25 and ATM protocols does not require that the path remain constant, this adds resiliencey at the cost of increasing the likelihood of having to deal with any out-of-sequence packets or any packet loss. TCP assigns a sequence number to each byte of data so that it may re-order data which have been put out of order because of fragmentation or re-transmission due to error or loss. Any application or protocol using TCP will always receive data in the correct order. The use of checksums allows detection of erroneous packets and TCP can then request re-transmission although this could be considered redundant in some cases as layer two protocols generally use more robust methods to ensure error free transmission between nodes; the TCP checksum is only for end to end error checking.

The TCP also includes mechanisms for flow control and congestion control. Flow control is the process of adjusting data rates such that a receiver does not become swamped with more data than it can process; it ensures that the sender will scale back the sending rate to allow the receiver to operate close to but not exceeding its limit. In a similar vein to flow control, congestion control seeks to control the rate of messages such that the network itself does not become congested and suffer collapse or performance degradation. By measuring the network between end points using the presence or lack of acknowledgement packets (packets which indicate the successful receipt of data packets at the receiver - ACK) some basic characteristics can be inferred. The rates of transmission can be adjusted by introducing delays. Congestion

control in TCP [25] (updated [26]) is a substantial topic in itself and there are a number of algorithms such as Vegas [27] [28] and NewReno [29] available.

As mentioned in the previous section, TCP, is perhaps the most commonly used layer four protocol in large networks and in the Internet. TCP cannot be partnered with any layer three protocols other than with the IP as they do not fully conform to the OSI model.

2.1.3.2 User Datagram Protocol

Because of the overhead that TCP can add to a packet stream and the delay it can introduce (if packets arrived badly out-of-order) it is not suitable for every type of application. The merits of reliable delivery have been discussed, but sometimes unreliable delivery can be justified. The user datagram protocol (UDP) [30] is a common counterpart to TCP when reliable delivery is not necessary. It is simpler than TCP and does not include algorithms to handle out-of-order packets or perform complex error checking or correction. UDP can perform a checksum on the header of its packets if requested, however, it is more normal to have these handled by a different layer. One disadvantage of UDP is the lack of flow and congestion control - because no acknowledgement of delivery is received by the sender, it cannot adjust its rate to match that of the receiver. It is possible that a UDP source could, inadvertently or otherwise, flood a network. Routers with packet-dropping queues can mitigate this to some extent.

Due to its simplicity UDP is useful in situations where the message size is small compared to the TCP overhead, where a limited latency/jitter is more important than ensuring no packet loss, such as in voice over internet protocol (VOIP), streaming transmission or online games, or where a large number of requests for a small volume of information exists, such as in domain name system (DNS) [31] [32] or dynamic host control protocol (DHCP) [33] requests. The ratio of UDP to TCP packets in an Internet backbone link has been observed to be between 0.11 and 0.2 [34]. However, this may be lower in a smaller network as the number of DNS or DHCP packets are likely to be higher.

2.1.4 Routing

The use of RPs was previously mentioned along with some background as to how they arose in circuit-switched networks as a manner in which to ensure connection between end users. In a packet-switched network, the same principle is applied albeit at an individual packet as

opposed to stream (call) level. The RP is concerned with finding a path through the network to establish an efficient and robust path between end users. It is generally assumed that in a wired network scenario the topology does not change rapidly, if at all, and that routing is somewhat simpler than in a wireless scenario where the network topology can be in a state of constant flux. The stability of wired routes makes routing a limited overhead (in terms of traffic) and as such most RPs are concerned with finding the best route through the network. The best route may be the shortest route although other constraints may be imposed, such as maximum delay, jitter and minimum bandwidth, with some protocols electing to choose the route with highest bandwidth. The study of RPs is a substantial topic in itself. However, some of the most prevalent for large networks are open shortest path first (OSPF) [35], intermediate system to intermediate system (IS-IS) [36], and border gateway protocol (BGP) [37], which is considered the core routing protocol for the Internet.

2.2 Overview of network measurement

The network statistic to measure, the location where it can best be measured, the scope and the mode of measurement are four defining characteristics of a measurement scheme for data networks. The first choice, what to measure, is determined by the user and generally chosen from the following:

Latency. Latency (or delay) is the elapsed time between a packet departing the relevant layer at the source node and its arrival at the corresponding layer of the destination node. Measurement can either be one-way or round-trip and if asymmetrical paths are assumed (i.e. the path between source and destination is not the reverse of the path between destination and source) then the round-trip time (RTT) is not necessarily twice the one-way delay. Standards for the reporting of these quantities and exacting requirements over their measurement are defined [38] in order that measurements can be made in non-homogeneous hardware environments.

Jitter. Jitter is the second order component of latency, i.e. jitter is the variance of the latency. It is useful as a metric of stability in a path, if jitter is high then the path latency may be unpredictable and not favourable to RPs which specify a desired level of latency.

Loss-rate. Loss-rate [39] is the ratio (although absolute values may be reported) of packets failing to successfully arrive at the destination divided by the number of packets sent by the source. A high loss-rate may be caused by packets exceeding their time to live (TTL) which

is used to restrict the forwarding of packets which have stayed too long in the network. The TTL is measured in seconds, and, in IP version 4 is decreased by each node which re-transmits the packet. In practise, each node must decrease the TTL by one unit. In IP version 6 it was renamed the hop limit to reflect current operation. Once the TTL of a packet reaches 0 it is discarded. This may be the case where a packet is undeliverable in a network. For example, the destination may no longer be part of the network or there may be a routing loop. Were there no TTL mechanism, undeliverable packets could swamp a network over time. A moderate loss-rate may be acceptable in situations such as VOIP where a packet arriving late will cause noticeable disruption to the quality of the call, but a lost packet may go un-noticed due to interpolation by the application.

Available bandwidth. The available bandwidth (ABW) is the minimum bandwidth that is available on a path (i.e. from source to destination). The ABW is used as a service level for QOS systems where limits are placed on most metrics. Applications such as VOIP or video-conferencing will often specify a minimum ABW necessary for acceptable operation to the end user. A decreasing ABW may indicate a link is becoming more *full* and as such it can be used as a load estimator. The measurement of ABW is more complex than might be imagined (and certainly more complex than latency measurement) and there are some varying approaches to the problem [40] [41] [42] [43].

The second choice, where to measure, is most often defined by the physical arrangement of the network infrastructure. In a large network or in the Internet, it may be the case that the devices or links are owned by different companies who are unwilling to share data about the performance of their networks. In this situation, statistical inference can be used to estimate from the available data (often data at end user nodes) some of the characteristics of the unavailable devices or links provided that some information about the topology and routing in the network is known.

The third choice, the scope of measurement, is also user defined; observation of a single path or link may be desired or a whole network may be under investigation. The volume of data to be collected and how it is to be collated from a distributed measurement scheme must be considered. Too high a volume of measurement traffic may distort the measurement results in an active measurement scheme.

Finally, the fourth choice, whether to use an active measurement scheme [44] which may send

test packets through the network to observe performance or a passive measurement scheme which observes data as it passes through the network and relies on the assumption that there will be enough traffic to form a robust estimate is a user choice as both are viable solutions in different situations.

Given the range of choices available, there exist a number of established tools which are designed to measure specific characteristics of the network in different situations [45] [46] [47] [48]. Note that any such list is almost certainly incomplete and that a fuller list can be found hosted by CAIDA [49] at http://www.caida.org/tools/taxonomy/performance.xml .

2.3 Inference methods for network measurement

Given the number of measurement choices available, what to measure, the scope, the mode and the location of measurement, it is understandable that there are many algorithms which have been published. This section presents an overview of some of the best known and characterised algorithms. This thesis concentrates on the use of inference methods for network measurement and in particular those of the *network tomography* type, a term used in [50] by Vardi although methods of projecting from one measurement of network data to another were published by Kruithof in 1937 [2] (a review, in English, is given by Krupp [51]). This short taxonomy concentrates on the characteristics measured, the scope and mode of measurement and the estimation/detection algorithms employed. The nomenclature used in describing each of the papers below is either that used in the paper itself or a variation of that used in the paper. The variation comes from a reduction in variables, for example, removing extraneous subscripts that may be un-necessary. The reason for retaining the nomenclature from the paper is for ease of comparison by the reader with the paper itself. Converting the mathematics to a common notation would add little value here.

Traffic intensity estimation. Vardi [50] estimates path traffic intensities (he uses the term source-destination (SD) traffic intensities) from link measurements using a maximum likelihood (ML) approach which is based on the method of moments (MOM) and utilises the expectation maximization (EM) algorithm [52] [53]. Vardi models a network as a fully connected set of nodes where there exists a direct path between any pair of nodes. The connections are known *a-priori* and are fixed. The number of pairs in the network (of n nodes) is therefore given by $c = n(n-1)/2$. It is assumed that the number of transmitted messages

(packets) between the elements of pair j in measurement period k is $X_j^{(k)} \sim \text{Poisson}(\lambda_j)$ for $j = 1 \ldots c, k = 1 \ldots K$. The vector $\mathbf{X}^k$ denotes the number of packet transmissions in measurement period k. $\mathbf{A}$ is defined to be a routing matrix (RM) (see the example in Section 2.4.1 for a description of routing matrices and Table 1.1 for the routing matrix for the above example). The network is measured on all links for time periods $k = 1 \ldots K$ giving rise to the vector $\mathbf{Y}^k = (Y_1^k, \ldots, Y_r^k)$ where r is the number of links in the network. The objective is therefore to determine the rates of the Poisson processes on each link given (2.1).

$$\mathbf{Y}^k = \mathbf{A} \cdot \mathbf{X}^k \tag{2.1}$$

Vardi then proceeds to solve this problem and thus determine the rates λ_j for two scenarios. The first assumes a fixed routing over the period of interest and the second assumes a random routing. The routing in the latter scenario is not truly random; packets transmitted between node pairs take a path determined by a fixed and known *a-priori* Markov chain specific to that pair.

In the fixed routing scenario, Vardi uses the ML method and compares this to a solution of the likelihood equations. Both give a unique but different answer. Further analysis of the solutions and their derivation shows that the maximum likelihood estimate (MLE) is, as expected, the more likely solution (in the statistical sense). Vardi then expands the approach to use the EM algorithm to search for a solution however he concludes that this is inefficient with respect to the computational effort required. He then proceeds to discuss approximating $\mathbf{Y}$ with a single Gaussian distribution (GA) for large K based upon the central limit theorem. This allows a reduction in computational effort as $\mathbf{Y}$ can be completely specified by its mean ($\mathbf{A}\lambda$) and covariance matrix ($\mathbf{A} \wedge \mathbf{A}'$) and leads to a solvable system of linear equations (in λ).

In the Markovian routing scenario Vardi uses similar methods but with increased complexity to allow for the effect of the non-fixed routing. This is particularly interesting as Vardi states that a fixed network is a special case of Markovian routing; most later papers (and this thesis) assume that the routing stays fixed, at least over the period of interest.

Link delay estimation. Similar to the work of Vardi, Cao *et al*, in [54] present an

origin-destination (OD)[2] network tomography algorithm. They seek to infer from link measurements (packet counts at routers), the packet counts between OD pairs. The authors make a similar assumption in that the distribution of a large number of packets can be approximated by a single Gaussian distribution reducing the complexity involved in finding and computing a solution. Specifically, they note in their equation (2) (reproduced here as (2.2)) that the link byte counts $\mathbf{y}_t$, observed for example at the entry to a router can be modelled as a routing matrix $\mathbf{A}$ multiplied by a representation of the OD byte counts $\mathbf{x}_t$. The mean is represented by λ which should not be confused with the rate of the Poisson process used by Vardi.

$$\mathbf{y}_t = \mathbf{A} \cdot \mathbf{x}_t \sim \mathcal{N}(\mathbf{A}\lambda, \mathbf{A}\mathbf{\Sigma}\mathbf{A}') \tag{2.2}$$

One note Cao *et al* make is of identifiability of estimates and give a condition that there is some relationship between the mean and the variance of the packet counts (which excludes them from being Gaussian). They define this condition in their equation (4) which is reproduced here as (2.3) where c is a constant.

$$\sigma^2(\lambda) = \lambda^c \tag{2.3}$$

Cao *et al* also comment on the suitability of using a continuous distribution such as the single Gaussian distribution to model a discrete quantity such as byte counts. Their assumption is that the range of the byte counts will be sufficiently high given the volume of traffic on the network that a continuous distribution will give a good enough approximation such that discreteness can be ignored.

One novel contribution of the paper is a moving time or local model for byte count estimation. The motivation is that this type of data is bursty with long periods of a low volume of data interleaved with short bursts of high activity. Thus (2.2) can be re-written as (2.4) for estimation at time t where the window size is $w = 2h + 1$ and h is the half-width. Cao *et al* note that as the windows are overlapping some smoothing of the data is implicit in this approach.

[2]OD is alternative nomenclature for SD; in this work SD is generally used although they are interchangeable unless noted.

$$\mathbf{y}_{t-h}, \ldots, \mathbf{y}_{t+h} \sim \mathcal{N}(\mathbf{A}\lambda_\mathbf{t}, \mathbf{A}\mathbf{\Sigma_t}\mathbf{A}') \tag{2.4}$$

Cao *et al* test their algorithms using real data (something not done by Vardi) and show they provide good accuracy. However, the simplistic local windowing scheme suffers from a loss of accuracy with choice of window size; larger windows provide too great a smoothing effect whilst smaller windows decrease the reliability of estimates.

Link delay inference. Coates & Nowak in [3] introduce the concept of measuring network link characteristics such as path delay distribution using end-to-end, path-based measurements. They suggest that an approach based on the MOM as one way to accomplish this. The system is modelled similarly to the above papers with a path defined as a connected set of links; the connection details being represented by the routing matrix. In a network with $j = 1 \ldots L$ links and $i = 1 \ldots P$ paths such that the routing matrix A is of size $P \times L$ then the delay of a packet transmitted across path i is as given in (2.5).

$$Y_i = \sum_{j \epsilon L} X_{ij} \tag{2.5}$$

The authors seek to estimate the cumulant generating function (CGF) of the packet delay distribution on each link, K_{X_j}. They begin by estimating the moment generating function of the packet delay distribution on each path $M_{Y_i}(t)$ as shown in (2.6).

$$\widehat{M_{Y_i}(t)} = \frac{1}{N} \sum_{k=1}^{N} e^{tY_{ik}} \tag{2.6}$$

Using the least squares (LS) algorithm (which is covered in more detail in Section 2.4.1) the estimated path moment generating function can be converted into an estimated link CGF; equation (2.7) shows how the CGF estimate for link j is formed.

$$\widehat{K_{X_j}} = \sum_{i=1}^{P} h_{ij} \cdot \log(\widehat{M_{Y_i}}) \tag{2.7}$$

Coates & Nowak proceed to consider the effects of bias on their estimators and offer a bias-correction scheme in an attempt to improve performance. The application suggested is

in finding the link with the highest delay in a network, i.e. bottleneck link detection. The method and application presented here is used throughout this thesis and is covered in more detail in Chapter 4. The work is extended by the authors in [1] which is a survey of tomography methods. In this, they postulate that tomography may be used to estimate characteristics other than delay distributions (such as loss rate) on network links from path-based measurements. They consider OD tomography which they describe as *essentially the antithesis of link-level network tomography*. The previously discussed papers provide a contrast with this work by tackling the problem of estimating one of Y or X in the same framework with almost identical system models.

Passive loss-rate inference. Tsang, Coates and Nowak in [55] demonstrate a method of using passive monitoring to infer link loss-rates from end-to-end measurement. This is of interest because of the use of *passive* monitoring - previous papers have used only active probing strategies. The authors assume that pairs of unicast packets transmitted with a short, fixed and known delay between each transmission will capture similar network statistics. They hypothesise that if the packets originate at the same node but have different destinations which have a mostly common route then, if one packet arrives but one is dropped (or lost) then it can be inferred that the packet was lost on part of the route that is not common to both. They do not comment on a strategy for choosing the packets but implicitly assume that a suitable volume of traffic will exist on each path in the network that pairs can be selected. They also do not comment on the spacing between packets other than to *assume that the timing between pairs of packets is considerably larger than the timing between two packets in each pair*.

They begin by defining α_i as the probability a packet is transmitted from node $p(i)$ to node i where $p(i)$ is the parent node of node i, i.e. the node upstream from node i which will transmit packets to node i. This is given in (2.8) and it can be then deduced that the probability of a packet being dropped on the link between $p(i)$ and i is $1 - \alpha_i$.

$$\alpha_i = \Pr\{\text{a packet is transmitted successfully from } p(i) \text{ to } i\} \tag{2.8}$$

The conditional probability that both packets in a packet pair successfully arrive at node i can then be defined as given in (2.9) and can be expected to be close to 1 if the packet spacing is low.

$$\gamma_i = \Pr\{\text{1st packet } p(i) \text{ to } i \mid \text{2nd packet } p(i) \text{ to } i\} \tag{2.9}$$

Tsang *et al* begin by considering the scenario of single packets flowing from a source S to a node i. They assume that n_i packets are sent and m_i received such that $n_i - m_i$ are dropped. Equation (2.10) then defines the likelihood of this as they assume the losses are Bernoulli.

$$l(m_i|n_i, p_i) = \left(\frac{n_i}{m_i}\right) p_i^{m_i}(1-p_i)^{n_i - m_i} \tag{2.10}$$

p_i is defined as the joint probability of the successful transmission of a packet on the path between S and i as given in (2.11) where $P(S,i)$ denotes the path between the source S and node i.

$$p_i = \prod_{j \epsilon P(S,i)} \alpha_j \tag{2.11}$$

The algorithm relies on sending packets from a source node S to two receiver nodes, i and j, which share a path with mostly common links which diverge at node $k_{i,j}$. S sends a large number of packet pairs where the first packet is destined for node i and the second for node j. $n_{i,j}$ denotes the number of packet pairs for which the second packet is successfully received at node j and $m_{i,j}$ denotes the number of packet pairs where both packets are successfully received at node i. $P(S,i)$ denotes the path between the source S and node i, $P(S,j)$ denotes the path between the source and node i and $P(S,k_{i,j})$ the path between the source S and the last common node on the path such that it defines the common subpath of i and j. The likelihood of $m_{i,j}$ given $n_{i,j}$ is given by (2.12) and (2.13).

$$l(m_{i,j}|n_{i,j}, p_{i,j}) = \left(\frac{n_{i,j}}{m_{i,j}}\right) p_{i,j}^{m_i,j}(1-p_{i,j})^{n_{i,j} - m_{i,j}} \tag{2.12}$$

$$p_{i,j} = \prod_{q \epsilon P(S,k_{i,j})} \gamma_q \cdot \prod_{z \epsilon P(k_{i,j})} \alpha_z \tag{2.13}$$

This sets the conditions for solving this problem using EM which the Tsang *et al* do. Using a

selection of simulation scenarios they find that the difference between the estimated and true loss rates is around 2%. The approach is very similar to the covariance based approaches in [5] and [56] although here the authors do not impose the condition of a strict binary tree which implies that it could be adapted to many network topologies.

Non-stationary delay inference; bottleneck link detection. Coates and Nowak's [57] approach to the inference of link delays using a sequential monte-carlo (SMC) method [58] is of particular interest as they demonstrate the ability to perform estimation in a non-stationary network. Many papers make the assumption that the network characteristics are stationary over the period of interest in order to perform bottleneck link detection. For the stationary case, the distribution of the queueing delay of packets on an individual network link are estimated empirically. M active probe packet pairs are sent through the network from source to receiver and indexed $m = 1 \dots M$ where $y_1(m)$ and $y_2(m)$ are the delays of the individual packets. The ordering of packet 1 and packet 2 is arbitrary. Coates and Nowak make the assumption that the delays are quantised such that each delay falls into a bin in the range $0, 1, \dots, K$ time units. A probability mass function (PMF) for each link can be estimated. Define $p_i = \{p_{i,0}, \dots, p_{i,K}\}$ which denotes the probabilities of a delay on link i having a delay of $0, 1, \dots, K$ time units. Defining $\mathbf{y} = \{y_1(m), y_2(m)\}$ Coates and Nowak are interested in finding the MLE of $\mathbf{p} = \{p_i\}$ the collection of all delay PMFs. The difficulty arises in finding the joint likelihood $l(\mathbf{y}|\mathbf{p})$ which is equal to the product of the individual likelihoods and cannot be determined analytically. As with other methods, the EM algorithm is used to find a numerical solution.

Coates and Nowak then proceed to consider a time-varying model. They define the time-varying distribution of measurement m with a window of size R as given in (2.14) where $z_i(l)$ is the unobserved delay experienced at queue i by measurement packets l and $1_{\{z_i(l)=j\}}$ is the indicator function for the event $\{z_i(l) = j\}$. $p_{i,R} = \{p_{i,j}(R, m)\}$ which denotes the time-varying probabilities on link i.

$$p_{i,j}(R, m) = \frac{1}{R} \sum_{l=m-R+1}^{m} 1_{\{z_i(l)=j\}} \tag{2.14}$$

The use of a time window to track a time varying change in delay was seen in [54]. The authors then proceed to develop the method to use a SMC approach to estimating the delay which will track more accurately the time-varying aspect. They also introduce a method which uses a particle filter to perform a similar task.

The assumption of stationarity in the data when considering an entrenched backbone wired network or fixed-link wireless network where the load is likely to remain constant over the period of interest is fair. However, this assumption does not hold when considering a network with rapidly changing conditions such as a network with a highly dynamic load caused for example by an aggregation of a small number of dynamic sources. In such scenarios, stationary methods would show reduced accuracy. Therefore the approaches presented are of interest as they attempt to address this more general and realistic problem.

Mixture models; multicast [59] covariance. Xia and Tse suggest in [5] that there are identifiability problems in using a single Gaussian distribution as an estimator for packet delay distributions. The authors define y as a vector of route (path) delays, x as a vector of link delays and R as the routing matrix such that the system is described as $Rx = y$. They consider modelling each link as a Gaussian random variable so that link i is modelled as X_i with mean μ_i and variance σ_i^2. If each link delay is assumed to be independent then the collection of L links can be considered multivariate Gaussian with mean vector $\mu = (\mu_1, \mu_2, \ldots, \mu_L)$ and covariance matrix $\Sigma = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_L^2)$. If $Y = RX$ then $Y = (Y_1, Y_2, \ldots, Y_M)$ is the vector representing the route attributes on M routes. Y is therefore multivariate Gaussian with mean $R\mu$ and covariance $\Sigma_Y = R\Sigma R'$.

To discount the Gaussian as a suitable estimator, the authors use the following proof. Denote P_θ as a multivariate Gaussian for the random vector Y with parameter θ where $\theta \epsilon \Theta$. Θ is the parameter space for θ and in the scenario above $\theta = (R\mu, \Sigma_Y)$. Their definition 3.1 states that for a parametric model to be identifiable $\theta_1 \neq \theta_2$ implies $P_{\theta_1} \neq P_{\theta_1}$ for all θ_1, θ_2. If $K = \mathrm{rank}\, R$ and $K < L$ then it is possible to select K routes such that any route vector is a linear combination of the selected K routes. It is then assumed that R is of size $K \times L$ with full row rank. Assuming that Σ_Y is invertible then the distribution is completely defined by its mean and covariance matrix and depends on μ through μR. Since $\mathrm{kernel}(R) \neq \{0\}$ there are vectors $\mu_1 \neq \mu_2$ such that $R\mu_1 = R\mu_2$. Thus, it may be impossible to separate the estimates for the two links.

Xia and Tse proceed to develop a similar case but modelling each link with a single exponential distribution. They consider the case of the moment generating function for a route comprised of exponentials and state that it will be different if one of the links is different and none of the parameters are identical. For example, consider two routes which diverge only at the last hop or two routes where the first is one hop longer than the second. It is then possible to eliminate

(i.e. determine) one parameter at a time in a binary tree network. They develop this method to form a density estimator for the route attribute (i.e. the path delay), finding an equation for the likelihood function and solving using EM. This is interesting because of the assumption of the binary tree topology which may not be valid in a real network. Xia and Tse present an approach to solving this problem using a moment estimating approach similar to MOM, but which has a strict requirement on the binary tree.

It should be noted that an approach to the problem using delay covariances was first shown in [56] by Duffield and Presti who, in the same paper, present some methods of topology inference. As mentioned in [5], the disadvantage of this method is that the probe packets used to measure the network must be multicast such that for each probe packet sent from the top (or root) node, each node at the end of the lowest branches receives 2^{n-1} probe packets, one for each route, where n is the depth of the tree.

Finally Xia and Tse expand their estimation from a single exponential distribution to an exponential mixture model using similar techniques showing some excellent fits to link distribution data.

Mixture models; link delay inference. Mixture models are also suggested by Shih and Hero [60] when they consider the inference of link delay distributions. They consider only lightly-loaded networks and are in agreement with Xia and Tse in that there may be problems identifying the means of individual elements within a Gaussian mixture model (GMM) from a joint probability density function (PDF). They proceed to develop a *hybrid discrete/continuous finite mixture model* by adding point-masses to the continuous components of the link delay PDF model. This is a break from the assumption of Cao [54] that a continuous distribution is sufficient to model the distribution and seems cumbersome - a particle filter method could replace the hybrid solution if a continuous solution was undesirable. They estimate the parameters of their models using a ML-EM algorithm which they extend to include penalty weighting in an attempt to improve accuracy.

Data windowing; link delay inference. Sun [6] presents work using the EM algorithm to solve a set of derived ML estimators in order to infer link delay distributions. This approach is similar to [60] in the use of ML-EM but here the authors suggest using a data windowing scheme to reduce the computational complexity by not having to decompose data multiple times (accomplished by building a database of results) and by not having to decompose data

which is not relevant at that point in time.

Fourier domain; link delay inference. Chen, Cao and Bu [61] take a different approach to manipulating distribution functions in their work where they infer link level delay distributions by manipulating characteristic functions instead of deriving ML equations. By working in the Fourier domain, the authors find it simpler to manipulate the characteristic functions and consider each link distribution to be best modelled by a finite mixture model. The model parameters are estimated using an iterative quadratic programming method which is similar to the EM algorithm.

The system is modelled mathematically in the normal manner $\mathbf{Y} = \mathbf{X} \cdot A$ where $\mathbf{Y}$ is a I dimensional vector of measurements, $\mathbf{X}$ is a J dimensional vector of network parameters and A an $I \times J$ routing matrix. Since $\mathbf{Y}$ can be considered the sum of independent components of $\mathbf{X}$ then it is possible to define the characteristic function of $\mathbf{Y}$ as given in (2.15) where A^j is the j^{th} column of A.

$$\phi_{\mathbf{Y}}(\mathbf{t}) = E[e^{i\mathbf{t}^T\mathbf{Y}}] = E[e^i \mathbf{t}^T AX] = \prod_{j=1}^{J} \phi_{X_j}(\mathbf{t}^T A^j) \tag{2.15}$$

The problem is that it is in general difficult to evaluate the distribution of Y because it is the result of a high order convolution. Consider next that each link distribution X_j is described by a probability density function $f_{X_j}(x_j; \theta_j)$ with the parameter θ_j unknown. $\theta = \{\theta_j;\, j = 1 \ldots J\}$ for J links. The joint characteristic function of $\mathbf{Y}$ can then be written as given in (2.16).

$$\phi_{\mathbf{Y}}(\mathbf{t}; \theta) = \phi_{\mathbf{X}}(A^T \mathbf{t}; \theta) \tag{2.16}$$

$$\hat{\phi}_{\mathbf{Y}}(\mathbf{t}; \theta) = \frac{1}{N} \sum_{n=1}^{N} \exp(i\mathbf{t}^T \mathbf{Y}(n)) \tag{2.17}$$

Using N sampled measurements, the empirical estimate of the joint characteristic function can be given by (2.17). It is necessary to minimise the L_2 distance between the model characteristic function and the empirical characteristic function under a probability measure μ as given in (2.18) and (2.19).

$$\hat{\theta} = \arg\min \int |\epsilon_N(\mathbf{t};\theta)|^2 \mathbf{d}\mu(\mathbf{t}) \tag{2.18}$$

$$\epsilon_N(\mathbf{t};\theta) = \sqrt{N}(\hat{\phi}_{\mathbf{Y}}(\mathbf{t}) - \phi_{\mathbf{Y}}(\mathbf{t};\theta)) \tag{2.19}$$

Chen *et al* proceed to solve this using a quadratic programming method and a weighted Monte-Carlo based approach. What is interesting is that they can change from a univariate model to a mixture model without much more difficulty, such is the flexibility of their method.

Compressed sensing; link delay inference. A method addressing a slightly different problem in the field of network tomography comes in a work by Coates, Pointurier and Rabbat [4]. In their paper, the authors apply the technique of compressed sensing (CS) to the problem of efficiently estimating the mean end-to-end delay on all paths in a network by monitoring only a subset of the paths. This differs from the approaches above which sought to estimate the delay (or other metric) on network links from measurements on network paths or *vice versa*.

The authors begin with the usual system model as given in (2.20). The network contains n_p paths and n_l links. $x^{(k)} \in \mathbb{R}^{n_l}$ is the vector of link-level performance metrics and $y^{(k)} \in \mathbb{R}^{n_p}$ is the vector of path-level performance metrics at time instant k. The routing matrix $G^{(k)}$ describes the interaction between links and paths with $G^{(k)}_{i,j} = 1$ if link j is part of path i.

$$y^{(k)} = G^{(j)} \cdot x^{(k)} \tag{2.20}$$

The authors then define that only a subset of all paths are observed. n_s is the number of metrics y_s observed on the subset s of all paths where $n_s = |s|$. Therefore the authors seek to infer the metrics on the remaining $n_p - n_s$ paths. The observations are not made by chance but based upon a selection matrix $A^{(k)} \in \{0,1\}^{n_s \times n_p}$ The presence of a 1 in the p^{th} column of $A^{(k)}$ indicates that the performance metric is observed on path p. It is then possible to define (2.21) the observed samples $y_s^{(k)} \in \mathbb{R}^{n_s}$.

$$y_s^{(k)} = A^{(k)} \cdot y^{(k)} \tag{2.21}$$

It is assumed that the routing matrix is fixed throughout the measurement process, although the

algorithm is adaptable to allow the routing matrix to change during the measurement period. Coates *et al* use diffusion wavelets to model the path level characteristics because they assume there is a high correlation between many of the paths, given that they can be constructed from similar links, i.e. the subpaths of some paths may be identical with paths differing by only one link. Clearly the choice of the selection matrix $A^{(k)}$ is critical and the authors provide an algorithm for accomplishing this based upon the decomposition of the routing matrix.

Their results are impressive and they state they can recover the mean network path delay using only 4% of the network paths (5 measurements per timestep in their example) with an error of less than 5%. Coates *et al* do however note that there is temporal correlation in the data and that the method could be improved to take account of this. The paper is interesting as most previous papers have not devoted much effort to the efficiency of measurement in the network.

2.4 General Concepts

Most inference based network tomography methods operate using the same principles and therefore there are concepts, theories and a nomenclature that are common to many of the methods examined in this thesis. The key algorithm is the LS algorithm and in the example below it is used in the conversion between path and link estimates. The problems in sampling traffic are discussed and some insight is given into how this may be performed. Also provided is a revision section on distribution functions which discusses some key facts and equations while the section on cumulant generating functions discusses some general methods of manipulation.

2.4.1 Least Squares

Coates and Nowak [3] introduced the concept of converting between path measurements and link estimates, i.e. inferring link characteristics from path measurements, using the pseudo-inverse of the network routing matrix to weight contributions from each path measurement to each link estimate.

The first step in this approach is the generation of the RM[3]. This is a matrix which describes how the links in the network are interconnected to form a set of paths. If an element in the

[3]The routing matrix is commonly called the routing table in physical devices. In this thesis the nomenclature routing matrix is preferred to emphasise how it can be manipulated using matrix theory.

matrix is one, this indicates that the link (the column) is part of the path (the row) that the element intersects; if the element is zero, the link is not part of the path. It is assumed that the routing matrix is of full rank or higher so that each link is sufficiently covered by paths as to draw an estimate from the combined measurements of more than one path. The nomenclature adopted for this thesis is to refer to the RM as H[4].

The next step is to invert the RM so that each element will indicate not whether a link is part of a path, but how much of the weight of a sample from a path, can be attributed to a link. This is based on the assumption that the delays on links are independant of delays on other links. The inversion type used is the L2 pseudo-inverse, more commonly called the Moore-Penrose pseudo-inverse [62] [63]. The inverse of the RM may not always exist as the RM may not be square or invertible hence the use of the pseudo-inverse, which is a more generalized version of the inverse. The pseudo-inverse will exist and be real for any RM since a RM cannot have non-real components. In this work, all RMs are of full, or higher, rank, and, in-keeping with other nomenclature, the pseudo-inverse is denoted as H^{-1}, with the element at row i and column j denoted as $h_{i,j}$.

The algorithm may be better understood by means of an example:

Observe in Figure 2.2 how H can describe the layout of the network; the row corresponding to path four (i.e., row four) has a 1 in the columns corresponding to links three and four (columns three and four). From study of the topology shown alongside, it can be seen that path four utilises only links three and four.

Equation (2.22) shows the L2 pseudo-inverse of the routing matrix in Figure 2.2 where, for example, $h_{3,4}$ is 0.5. Thus, an estimate of link one L_1 would be constructed as shown in equation (2.23), where P_x is the measurement on path x. Because of the transform, entries in H^{-1} are not necessarily integers.

$$H^{-1} = \begin{bmatrix} 0.5 & 0.5 & 0 & -0.5 & -0.5 \\ 0.25 & -0.25 & 0 & 0.25 & 0.75 \\ -0.5 & -0.5 & 1 & 0.5 & 0.5 \\ 0.25 & 0.75 & -1 & 0.25 & -0.25 \end{bmatrix} \tag{2.22}$$

[4] *cf* The usual depiction for a channel modelling matrix in wireless communications.

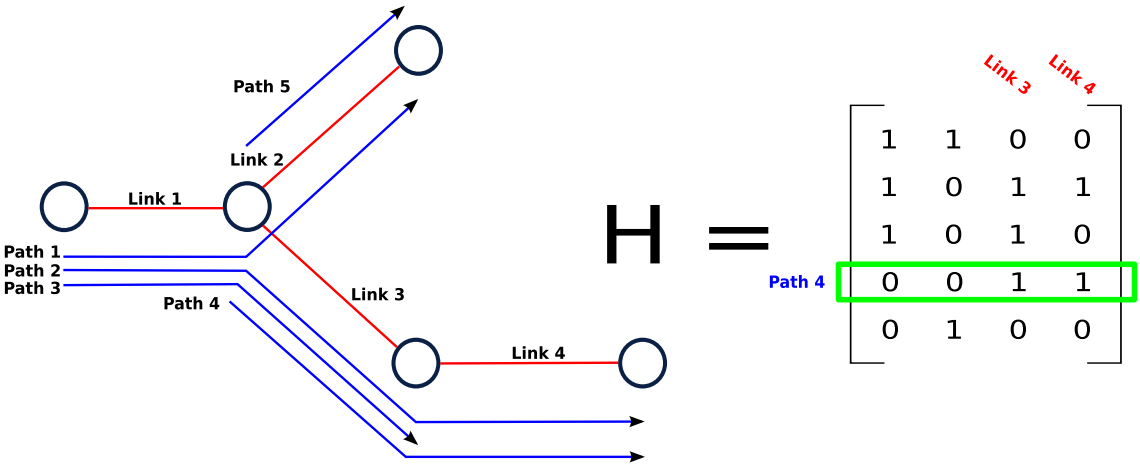


Figure 2.2: *An example illustrating how H describes the layout of the network. The row for path four has a* 1 *in the columns corresponding to links three and four indicating that path four uses only links three and four; this can be seen in the topology shown on the left-hand side.*

$$\begin{aligned} L_1 &= h_{1,1} \times P_1 + h_{1,2} \times P_2 + h_{1,3} \times P_3 + h_{1,4} \times P_4 + h_{1,5} \times P_5 \\ L_1 &= 0.5 \times P_1 + 0.5 \times P_2 + 0 \times P_3 - 0.5 \times P_4 - 0.5 \times P_5 \end{aligned} \tag{2.23}$$

There is one problem not mentioned by Coates & Nowak when using this approach - there are contributions towards link estimates that come from paths which the link is not part of. This is quite unusual as it implies an erroneous measurement on one path may have an effect on an unconnected link. This is unlikely to happen in reality as the distributed nature of a packet-switched network and RPs stop problems in one part of the network negatively affecting another part. One method to alleviate this problem and therefore increase the reliability of the estimates is to use a different method of generating H^{-1} from H. The L1 pseudo-inverse [64] is a candidate for this task however it is not examined in this work. The use of the L2 algorithm is retained as it allows a performance comparison between our methods and published methods.

2.4.2 Sampling

In the context of network measurement, sampling refers to the process of determining some information from some metric of interest about the packets on the network. The methods by which the sampling may be performed are examined here in the context of an active probing

system, however, the concepts are similarly applicable to a passive monitoring system. The inverse of sampling, recovering some information about the original data from the sampled data, is also an interesting problem which is given some attention in [65].

Perhaps the best known method takes advantage of a theory called Poisson Arrivals See Time Average PASTA formalised in 1982 by Wolff [66]. Poisson arrivals see time average (PASTA) is described in [67] as *observations made of a system at time instants obeying a Poisson process, when averaged, converge to give the 'true' value, that is to the average that an ideal observer would make when monitoring the system continuously over time*.

In this context, one of measuring packet delays [68] in a network, a Poisson process can be defined as one in which packets are sent into a network [69] with an inter-packet temporal spacing which is drawn from an exponential distribution with rate λ. Consider that packets are injected at some time, t, and the delay they experience is caused by queueing at a device, i.e. a router. The number of packets in said queue can be likened to the state of the network, $N(t)$, at time t; define $Pr(N(t)) = k$ as the probability of there being k packets in the queue at time t. The packets injected into the network will experience a delay based upon the number of packets in the queue. Assume that the inter-packet spacing is independent of previous spacings. Also assume that the packets injected affect the number of packets in the queue in some way, i.e. by adding to it, but the manner of this affect is not defined. Now define P_k as the steady state probability that there are k packets in the queue as given in (2.24).

$$P_k = \lim_{t \to \infty} Pr\{N(t) = k\} \tag{2.24}$$

Define A_k to be the probability that the Poisson process observes k packets in the queue at time t and that a new packet arrives in time interval $(t, t + \Delta)$. Therefore, using the method from [70]:

$$
\begin{aligned}
A_k &= \lim_{t\to\infty} \Pr\{N(t) = k \mid \text{ an arrival appears in } (t, t+\Delta t)\} \\
&= \lim_{t\to\infty} \lim_{\Delta\to 0} \frac{\Pr\{N(t) = k, \text{ an arrival appears in } (t, t+\Delta t)\}}{\Pr\{\text{ an arrival appears in } (t, t+\Delta t)\}} \\
&= \lim_{t\to\infty} \lim_{\Delta\to 0} \frac{\Pr\{N(t) = k\}\Pr\{\text{ an arrival appears in } (t, t+\Delta t) \mid N(t) = k\}}{\Pr\{\text{ an arrival appears in } (t, t+\Delta t)\}} \\
&= \lim_{t\to\infty} \lim_{\Delta\to 0} \frac{\Pr\{N(t) = k\}\Pr\{\text{ an arrival appears in } (t, t+\Delta t)\}}{\Pr\{\text{ an arrival appears in } (t, t+\Delta t)\}} \\
&= \lim_{t\to\infty} \lim_{\Delta\to 0} \Pr\{N(t) = k\} \\
&= \lim_{t\to\infty} \Pr\{N(t) = k\} \\
&= P_k
\end{aligned} \tag{2.25}
$$

The implication of this theorem is that if the measurement of packet delay (or any other metric of interest) in the network is done not on a periodic basis but at periods which are determined by a Poisson process then this figure should be identical to the figure which would have been gained had the system been measured for all time. As an example, consider that the Poisson process records a delay of 5ms on 14% of packets. PASTA allows extrapolation of this figure to say that 5ms will be the delay of 14% of all packets in the network, not just those measured.

The most common measurement made using this theory is of packet delay on paths but with care it could be used to justify the measurement of many characteristics. Tariq et al. [71] argue that whilst PASTA is justifiable as a method to sample network traffic, the use of a period process in generating the probe packets can yield results which are equally accurate and valid [72] [73]. They take as an example the measurement of median RTTs using both Poisson and periodic processes to generate probing traffic. Their conclusion is that periodic probes can, in cases where the network traffic is similar to that of the Internet or a large, high traffic network, i.e. aggregated from multiple sources, produce results equal to that of PASTA. However, they do note that periodic probe traffic is unable to highlight any behaviour which occurs on a timescale smaller than the inter-probe time spacing, this is effectively aliasing. In [74] synchronisation of periodic packets and probes is also considered. These issues are important because many network phenomena are short lived and bursty in nature. Perhaps the most interesting observation from Floyd and Jacobson is that if probe packets are dispatched from the source with a Poisson generated spacing their arrival times at a particular destination will be subject to some modification by the network and therefore the spacing may not conform

to the Poisson distribution [73]: in essence they imply the PASTA property will not apply.

It would appear that probe traffic can be of either the periodic or Poisson type without loss of information, subject to the constraint of ensuring the inter-packet spacing is of the same order as the phenomena to be observed.

2.4.3 Burstiness of network traffic

As mentioned above, traffic and phenomena on a network can appear bursty in nature. The study and explanations for the behaviour is a substantial study in itself so what follows is a brief description. The behaviour originates with the underlying distributions driving network traffic sources. These tend to be ON/OFF sources with the ON and OFF periods having a heavy-tailed distribution. Aggregation of heavy-tailed sources forms time-series which are self-similar [75] and the similarity can be observed in web traffic [76]. Network phenomena tend to appear in clusters and the aggregation process tends to amplify this clustering process rather than reduce it. The opposite, a smoothing of the time series, would be expected with, for example, a Poisson based underlying process. [73] notes that Poisson processes, which have traditionally been used to model traffic, are poor models. In a self-similar process, the time-series data will appear similar at different scales. Self-similar processes exhibit long range dependance. The degree of self-similarity can be measured by the Hurst parameter and in [75] this is used as a metric for the *burstiness* of the data. Long range dependance implies that, for example, the sizes of packets transferred across a network are not strictly independant.

2.4.4 Distribution functions

In this thesis working with the the cumulative distribution function (CDF) of a distribution is generally preferred. The CDF is gauranteed to exist at all points for any function and is bounded by $[0, 1]$. The PDF does not exist in all cases. The relationship between the PDF and the CDF is that the PDF, where it exists, is the derivative of the CDF, as shown below.

$$\mathrm{PDF}(x) = \frac{d}{dx}\mathrm{CDF}(x) \tag{2.26}$$

This relationship is one of the reasons why the CDF is favoured over the PDF; it gives a means to determine the weight of the PDF *yet to come* at a value of the CDF parameter. More formally,

the CDF is the probability that a random variable, X, takes a value less than or equal to x.

$$\mathrm{CDF}(x) = \mathrm{P}(X \leq x) \tag{2.27}$$

Related to this and sometimes more useful is the complimentary cumulative distribution function (CCDF).

$$\mathrm{CDF}(x) = \mathrm{P}(X > x) = 1 - \mathrm{CDF}(x) \tag{2.28}$$

Comparisons of CDFs are extremely useful in determining the fit of an empirical CDF (such as that gained from a measured data set) and an ideal distribution (such as might be considered as a hypothesis about the nature of the distribution of the data). The Kolmogorov-Smirnov (KS) test is designed to do exactly this: to give a numerical comparison of two continuous CDFs. An analogous, and perhaps more flexible test, can be performed with the Kullback-Leibler divergence (KLD) which gives a measure of the dis-similarity of two PDFs (or PMFs in the discrete case). The KLD comes from a family of divergences, the f-divergence (FD) of which the basic form is a cumulative sum of the convex function of the ratio of the two PDFs evaluated at a set of discrete points; see Equation 2.29 where PDF_1 and PDF_2 are defined on A.

$$\mathrm{FD}(A) = \sum_{a \in A} \text{convex fn} \left(\frac{\mathrm{PDF}_1(\mathrm{a})}{\mathrm{PDF}_2(\mathrm{a})} \right) \tag{2.29}$$

2.4.5 Cumulant generating functions

It will also be seen later that the cumulant generating function (CGF) is another method of describing a data distribution. It is possible to convert between moments (which may be more simple to estimate) and cumulants using a recurrence relation as shown in Equation 2.30 where κ_n represents the n^{th} cumulant and μ'_n the n^{th} moment.

$$\kappa_n = \mu'_n - \sum_{k=1}^{n-1} \binom{n-1}{k-1} \kappa_k \mu'_{n-k} \tag{2.30}$$

Describing data with a CGF is a similar approach to using a parametric distribution but without

a restriction on the number of parameters: given a large enough number of cumulants, it is possible to describe any arbitrary distribution. The advantage of using a CGF is that it is not necessary to make an *a-priori* decision about the characteristics of the data to be described. In the conext of network tomography, this makes it possible to construct, when combined with LS, an equation for a link distribution (the desired information) in terms of the inverse routing matrix and the cumulants of the path data (the measured information). The problem with using a CGF is in comparing it directly to a CDF or PDF: conversion from one form to the other is complex and may not always be calculable.

One method of conversion is to make use of the Gram-Charlier A series (GCA) [77] in which a Gaussian PDF is distorted by multiplication by the Hermite polynomial series to give an approximation to the desired distribution [78]. Unfortunately, GCA is not guaranteed to converge but a similar and related series is: the Edgeworth series [79].

In Edgeworth and in the GCA, the weights attached to the terms in the Hermite series are related to the cumulants of the data thus linking a PDF and measured cumulants. Cramér [80] gives the formula for the Edgeworth series as shown in Equation 2.31 for the first four cumulants (although the series is infinite) where $q^{(z)}(x)$ is the z-th derivative of q (which is assumed to be Gaussian) with respect to x; μ_p is the p-th moment and σ is the standard deviation.

$$f(x) = q(x) - \frac{1}{3!}\frac{\mu_3}{\sigma^3}q^{(3)}(x) + \frac{1}{4!}(\frac{\mu_4}{\sigma_4} - 3)q^{(4)}(x) \tag{2.31}$$

Unfortunately, in practice it is found that the distortion factor introduced by the Hermite polynomial is not large enough to give a good approximation to the empirical distribution, even when using 20 terms. One workaround that is used in this thesis is to estimate the link CGF (using a suitable estimator and LS) and then map a restricted number of cumulants onto a parametric distribution. This is not ideal because parametric distributions tend to be limited to four or fewer moments making it impossible to exploit the higher order statistics of the data.

2.5 Summary

In this chapter an overview of large-scale data networks and network tomography has been presented. It was seen that a packet-switched network can, by the use of virtual circuit techniques be made to resemble a circuit-switched network. The protocols in use on

packet-switched networks for both reliable and unreliable transmission were also discussed. The OSI model of networks was introduced and it was seen that this allows abstraction away from hardware-specific methods to enable discussion of generic networks at a specific layer without cnonsideration of the underlying systems.

Some common measurement techniques and algorithms were summarised in a short taxonomy which focused on the estimation and inference mechanisms used. Each algorithm was observed to have its advantages but they are designed for specific tasks and generalisation is not always possible. Finally, there was a review of some general concepts and nomenclature which are relevant to work later in this thesis.

Chapter 3
Simulation of computer networks

Introduction

Two commonly used methods exist for research into the operation of computer networks: the first is to physically construct the network one wishes to observe and perform direct measurements on any parts of interest, the second is to simulate the network and perform virtual measurements on any parts of interest. The former offers an absolute answer, the observations will be real but the cost of implementing and running an experiment may be high. The latter offers a lower cost per experiment but the correctness of any observations is dependant on the quality of the simulation tool used. There is a third method, emulation, which comes partway between the two; however its use is limited and it is not discussed further.

As was previously hinted, simulation is used for the work in this thesis; specifically the network simulator version 2 (ns2) [81] which is a commonly used, open source network simulator. Other free simulators such as Scalable Self-Organizing Simulation (S3) and GloMoSim are available and offer much of the same functionality as ns2; a commercial offering, OpNET is possibly the closest to ns2 in terms of interoperability.

3.1 Simulation methodology

The simulations used in this thesis are composed of three broad steps: simulation of the network (ns2), extraction of data from the ns2 output (AWK) and processing and plotting of the data (MATLAB). Each step may be composed of more than one sub-step; this is especially true for the AWK data extraction step which may require multiple iterations or staged processing of data.

3.2 Capabilities and limitations of ns2

ns2 is an event-driven simulator which means that the simulator operates using a long sequence of events. Initially, the sequence lists every event that is scheduled in the network and the time at which it is going to happen; these events are usually defined by the user. ns2 executes events starting at time 0 and proceeding through the sequence, executing each event as it finds it. An event may schedule future events. For example, at time 3, a packet source may begin generating packets, there will be a delay and then the packets will be transmitted, received, decoded, processed etc. Events are added to the event sequence at the relevant time, i.e., if a packet was transmitted at time 7 and the delay before it arrived at the receiver was 3 then the event corresponding to the receipt would be scheduled for time 10. ns2 processes one event at a time to keep complexity low as parallelisation could lead to conflicts.

The advantage of this method is that periods of time in which no events occur do not have to be simulated; computational effort is only expended for events. Timings for events, i.e. how long it takes for a packet to get from layer X to layer Y, are pre-defined (based on real-world knowledge and measurements) and can be adjusted if necessary[1]. One criticism often incorrectly levelled at ns2 is the inability to deal with concurrent events however, this is not the case. Two events can be scheduled to occur at the same time (since the time resolution is of the order of microseconds), however one will be processed before the other (sticking to the process one event at a time rule) - the order of processing only becomes important if both events seek to modify the same item (packet, node etc) at the same time. If this does happen then anomalous behaviour may be seen. This is due to bad implementation by the user and mirrors what would happen in a real-world scenario. Consider two pieces of code which may either power on or power off a device which may be initiated by an external interrupt. Both could be triggered very close together (in time) such that the instant of switching the power state will be the same. In this case the hardware would have to resolve the problem and in a simulation the model would do likewise. It should be noted that the particular method in which such cases are resolved will be different. The design choices made in the hardware scenario are likely to be more constrained than those made in the simulation scenario.

ns2 fully implements many common networking protocols (such as TCP), medium access control layer (MAC) protocols, physical layer (PHY) protocols and both wired and wireless

[1]None of these timings are modified from the default in this thesis.

transmission media. It is mainly used for research at the lower layers, for example, the transport layer and below (following the OSI), and generally has more support for protocols operating at those layers than at the higher layers.

3.3 Network simulation using ns2

The ns2 stage of the simulation has two inputs and one output. The inputs to the simulator are a network description and a traffic description. The output is a trace file.

The network description gives the specification of the infrastructure of the network. It defines the characteristics of entities such as nodes and links. A node description includes a model for the link layer (LL), PHY, MAC, energy constraints, mobility model and so forth. Links are similarly defined giving details of the nodes they connect, the bandwidth, the delay, and the queue type. Part of a typical network description can be seen below.

```
1 $ns duplex-link $n0 $n1 1Mb 250ms DropTail # Link 1
2 $ns duplex-link $n1 $n2 1Mb 100ms DropTail # Link 2
3 $ns duplex-link $n1 $n3 1Mb  80ms DropTail # Link 3
4 $ns duplex-link $n3 $n4 1Mb  10ms DropTail # Link 4
```

This description indicates that there are four, bi-directional links (`duplex-link`) connecting 5 nodes (`$n0` through `$n4`) with a 1Mb bandwidth and a droptail queue. Each link has a delay associated with it which is added to each packet traversing the link, for example link 1 has a delay of 250ms. Since this is a wired simulation, there is no specification required of the nodes themselves; the assumption is that they are of the default type and remain active over the whole simulation.

The traffic description gives the specification of each traffic source and sink in the network. It attaches sources and sinks[2] to nodes as defined in the network description and allocates a type to each, i.e. TCP, UDP or NULL (for sinks). Routes are defined between a source and a sink so that traffic has a path to follow - it should be noted that only the end-points are given, it is up to the RP to determine how to transport the traffic between the nodes. Attached to a route is an application such as CBR or Pareto - these are the algorithms which generate the packets and include statistical measures which determine when packets are sent (which are generally

[2]Sources and sinks are gathered in one class, Agent, in the ns2 nomenclature.

specified by the user).

An example of a traffic description is given below:

```
1 set udp_(10) [new Agent/UDP]
2 $ns attach-agent $n0 $udp_(10)
3 set null_(10) [new Agent/Null]
4 $ns attach-agent $n1 $null_(10)
5 set par_(10) [new Application/Traffic/Pareto]
6 $par_(10) set packetSize_ 512
7 $par_(10) set burst_time_ 500ms
8 $par_(10) set idle_time_ 500ms
9 $par_(10) set rate_ 400k
10 $par_(10) set shape_ 1.5
11 $par_(10) attach-agent $udp_(10)
12 $ns connect $udp_(10) $null_(10)
13 $ns at 0.0 "$par_(10) start"
```

The first line creates a UDP agent, a packet source which uses UDP at the transport layer and the second line attaches this agent to node 0. The third and fourth lines are similar, creating a null agent (a packet sink) and attaching it to node 1. On line 5 an application is created. This is the function which will generate the packets to be transported and in this case the type is Pareto implying that all packets generated will have a temporal distribution based on the Pareto function. The next 5 lines give the details of the distribution (burst time equal to idle time, shape factor), the size of the packets (512 bytes) and the rate (400kb/s). The 11th line attaches this generator to the UDP agent and the 12th connects the packet source and sink forming a virtual link between the nodes. In the final line, ns2 is instructed to start the Pareto generator at time 0, i.e. when the simulation starts. No finish time is explicitly defined here as a global finish instruction for all traffic is issued elsewhere.

A typical trace file can contain a large volume of data. By default ns2 logs every transaction that occurs in the network which may be too much, especially for a wireless scenario which includes moving nodes. It is possible to reduce the verbosity of the output by logging only transactions which will be used in later analysis, for example, if the interest is only in events happening at the agent layer then logging MAC and PHY layer transactions serves little purpose. Shown below is a segment of a trace file for a wired network - these traces are generally much less complex than those for wireless networks.

```
1 + 0 3 4 tcp 40 ------- 0 3.5 4.3 0 4
2 - 0 3 4 tcp 40 ------- 0 3.5 4.3 0 4
3 + 0 3 4 tcp 40 ------- 0 3.6 4.4 0 5
4 + 0 3 4 tcp 40 ------- 0 3.7 4.5 0 6
5 - 0.00032 0 1 tcp 40 ------- 0 0.5 1.4 0 1
6 - 0.00032 3 4 tcp 40 ------- 0 3.6 4.4 0 5
7 - 0.00064 3 4 tcp 40 ------- 0 3.7 4.5 0 6
8 r 0.01032 3 4 tcp 40 ------- 0 3.5 4.3 0 4
```

The first column describes the event occurring from a selection of five possibilities: enqueue a packet in the send buffer (`+`), dequeue a packet from the send buffer (`-`), receive at packet the agent layer (`r`), drop a packet (`d`) or error (`e`). The second column is the timestamp of this event relative to 0, the start of the simulation. The third and fourth columns are the source and destination for this hop of the journey respectively (a packet may traverse many links en route from source to destination). The fifth column is the traffic application type with the sixth column representing the packet size in bytes. Column seven is reserved for any special flags, a dash indicating nothing set whilst column eight is the flow identifier. Columns nine and ten are the source and destination addresses, respectively, of the packet for the whole route (as opposed to columns three and four which are for the current hop only). The part preceding the period is the node address and the part after is the port; the use of port numbers allows multiple packet streams between a pair of nodes. Finally, column eleven is the sequence number and column twelve is the unique packet identifier.

Using packet 4 as an example: on line 1 it can be seen that it is travelling from node 3 to node 4, it is a TCP packet of 40 bytes with no flags set and it is placed on the send buffer at time 0. The source and destination addresses for this hop are equal to that for the route so the packet is only travelling one hop. On line 2 it is dequeued, also at time 0, and sent. The packet reaches node 4 on line 8 at time 0.01032, total delay 0.01032s.

3.4 Data extraction and conversion

Once the trace file has been obtained it is likely to contain *excess* information, i.e. data that could not be filtered out from ns2 and which is not necessary in this scenario. An example would be data about a node which, while essential to the simulation as part of the network infrastructure, is not under examination. Thus, it is necessary to pre-process the trace file into

a more terse version; an advantage of this stage is that reducing the size of the trace file also reduces the length of time required to process any algorithms using it. There are a choice of tools available - given that the trace file is text-based, common UNIX utilities such as `grep`, `awk`, `perl` or `sed` may be used. awk appears to be the most commonly used and it is also the tool of choice in this thesis. Weak typing makes it flexible enough to handle filtering on a range of alphanumeric data such as node names, link numbers or time.

awk operates using rulesets, normally supplied in a file, and operates on whole input files. Rulesets are general filters and although they may contain mathematical functions, all but the most basic are avoided at this stage leaving the heavy mathematical processing for MATLAB. A basic example of a filter-type ruleset is seen below.

```
$1=="+" || $1=="r" {print}
```

This filter tells awk to print out only lines where the first column (`$1`) contains `+` or `r`. The implication is that the output from awk will only contain those lines where a packet has been generated by a node at the agent layer and enqueued for transmission (+) or received by a node at the agent layer (r). Since there is interest only in these events, this filter is used as a pre-processor. To use the earlier traffic segment as an example, below is the same segment after pre-processing (blank lines are shown to retain line numbering for reference). It is possible to add a further condition to the ruleset above, such as `$5==''cbr''` which would consider only packets of the type constant bit-rate (CBR). Later in this thesis, this filter is used to isolate probe packets from other traffic on a link or path.

```
1 + 0 3 4 tcp 40 ------- 0 3.5 4.3 0 4
2
3 + 0 3 4 tcp 40 ------- 0 3.6 4.4 0 5
4 + 0 3 4 tcp 40 ------- 0 3.7 4.5 0 6
5
6
7
8 r 0.01032 3 4 tcp 40 ------- 0 3.5 4.3 0 4
```

The goal here is in calculating the delay of packets and awk again provides a method of doing this. As the trace file now only contains the transmission and reception data, it is possible to calculate delay at this point using a ruleset as shown below.

```
1 $1=="+" && $3==src && substr($9,1,1)==src && substr($10,1,1)==dst
```

```
2 { x=$12; send[x]=$2; }
3
4 $1=="r" && $4==dst && substr($9,1,1)==src && substr($10,1,1)==dst
5 { y=$12; recv[y]=$2; }
```

The first condition (line 1) can be interpreted as searching for lines where column 1 is a `+`, column 3 matches the variable `src` (passed in from the calling script), the first character of column 9 also matches the variable `src` and the first character of column 10 matches the variable `dst` (also passed in). The checking of `src` and `dst` ensures that the filter is isolating packets originating at the current node and which are intended for the desired destination; this stops examination of packets which are at the current node but *en route* to the destination as only end-to-end delays are of interest. The second line of the stanza (contained with the curly braces) tells awk what to do if the condition matches; in the case of the first stanza, it assigns variable `x` the value of the 12th column (the unique packet id) then uses this as an index for the array `send` which contains, at index `x` the timestamp of the packet.

The second stanza (lines 4 and 5) performs a similar function but for the reception of packets. What both imply is that two arrays (`send` and `recv`) are formed which are indexed by the packet identifier and contain timestamps of packets. The delay is computed by subtracting values in `send` from the values at the same index in `recv`.

Using the previous example data segment, the two arrays would contain the following:

```
1 send[4]=0       recv[4]=0.01302
2 send[5]=0       recv[5]=
3 send[6]=0       recv[6]=
```

The entries marked as blank (recv[5] and recv[6]) are a result of not initialising the arrays to a known value, something not necessary in awk. Before computing the delay values, it is necessary to test that send[x] and recv[x] are not undefined. This step is shown below.

```
1 for ( x in send ) {
2   if (recv[x] != 0 && send[x] != 0) {
3       printf "%.6f, ", recv[x] - send[x];
4   }
5 }
```

The test `recv[x]!=0 && send[x]!=0` checks to see that values exist. Awk implicitly

assigns a value of 0 to the unassigned elements but does not use this value, i.e. if asked to print out an unassigned value it will leave it blank as seen above, but it does allow it to be tested against. The output is a fixed format value of delay with 6 figures after the decimal point as the output from this process is designed to be fed as an input to MATLAB.

Windowing

Whilst the above describes the procedure for dealing with cases where the desire is to gather *all* delays from a trace file, it is also possible to add a filtering mechanism to compute delay across a *windowed* subset of the trace file. It shall be seen later that is necessary for some types of algorithm and so it is discussed here.

Firstly, it is necessary to isolate the time window of data of interest from the whole trace. Given that a typical trace file for a 5000 second simulation (not at all uncommon) is around 1.1GB, it is necessary to employ some useful awk tricks to reduce the time necessary for parsing such a trace. The time taken parsing large traces is due to awk's necessity to search a trace line by line from the first line, if the data is towards the end of the file, or, if an algorithm requires sequential parts of a file, this will be a slow process. Assuming that the original trace file has been pre-processed to contain only enqueue and receipt actions on probe packets (or any packets of interest), it is possible to use a filter based upon the code shown below.

```
1 {getline NR < "a.prb"}
2 $2<=(t*resln) && $2>((t-1)*resln) {print}
3
4 # This clever little line stops us parsing the rest of the file
5 $2>(t*resln) {print NR > "a.prb"; exit}
```

The first line reads from a file `a.prb` the line number on which to begin the search. This makes it possible to skip quickly to a part of the file if it is known that the earlier lines have been searched (i.e. on a previous iteration). Since the trace file is time ordered, this is valid. Any lines where the timestamp is between `(t-1)*resln` and `t*resln` are then printed. `t` is the variable containing the block number that is under examination and `resln` is the time resolution. For example, if the desire is to have blocks of 20 seconds of data and the wish is to examine the data in the third block, `resln` would be set to be 20 and `t` to 3. The above code would then output all packets with timestamps between 40s and 60s.

Once the windowed dataset has been extracted, it can then be processed to extract any metrics of interest such as delay as discussed above.

3.5 Signal processing and data representation

It would be entirely possible to perform the signal processing part of the chain in awk, however, MATLAB is a pragmatic choice. Its vector based operation and large library of common functions reduces the difficulty and possible errors in implementing complex algorithms. As an example, the code used in Chapter 4 for computing the GA algorithm, as shown in Algorithm 1, is shown below.

```
function[w1 w2 w3 w4]=ga2(data1, data2, data3, data4, data5, x, h)

% Fit normal to Y
[norm_p1(1), norm_p1(2)]=normfit(data1);
[norm_p2(1), norm_p2(2)]=normfit(data2);
[norm_p3(1), norm_p3(2)]=normfit(data3);
[norm_p4(1), norm_p4(2)]=normfit(data4);
[norm_p5(1), norm_p5(2)]=normfit(data5);

% Convert params to array
param_a=[norm_p1(1) norm_p2(1) norm_p3(1) norm_p4(1) norm_p5(1)];
param_b=[norm_p1(2)^2 norm_p2(2)^2 norm_p3(2)^2 ...
                                    norm_p4(2)^2 norm_p5(2)^2];

% Transform params
norm_p1_t(1)=h(1,:)*param_a';
norm_p2_t(1)=h(2,:)*param_a';
norm_p3_t(1)=h(3,:)*param_a';
norm_p4_t(1)=h(4,:)*param_a';

h2=h.*h;

norm_p1_t(2)=h2(1,:)*param_b';
norm_p2_t(2)=h2(2,:)*param_b';
norm_p3_t(2)=h2(3,:)*param_b';
norm_p4_t(2)=h2(4,:)*param_b';
```

```
27
28 % Generate normal cdf at x for X
29 w1=normcdf(x,norm_p1_t(1),sqrt(norm_p1_t(2)));
30 w2=normcdf(x,norm_p2_t(1),sqrt(norm_p2_t(2)));
31 w3=normcdf(x,norm_p3_t(1),sqrt(norm_p3_t(2)));
32 w4=normcdf(x,norm_p4_t(1),sqrt(norm_p4_t(2)));
33
34 end
```

Line 1 is the function definition. There are seven inputs, the five `data` vectors which are vectors of delay values obtained from the awk processing step, `x` which is a vector respresenting the CDF over which the estimated delay distribution should be computed and `h`, the matrix representing the inverse routing matrix. There are four outputs, `w1` through `w4` which are the estimated delay distributions. In-keeping with the nomenclature, the statistics of the five data vectors are considered the *Y* data and the estimated output statistics *X*.

Lines 4 to 9 show the use of the inbuilt `normfit` function which estimates the mean and standard variation of the data passed to it. The results for each Y vector are stored in another vector with the first element, i.e. `norm_p1(1)`, containing the mean and the second element i.e., `norm_p1(2)`, containing the standard deviation. Lines 11 and 12 group these elements into two larger vectors, `param_a` for the means and `param_b` for the variances. In the case of the latter, it is necessary to convert from standard deviation to variance.

Lines 16 to 19 deal with the conversion process, using the LS method, between Y and X data. Lines 23 to 26 perform the same function but for the variances. Since variance is treated as a noise process, it is necessary to square `h` on line 21.

Finally, lines 29 to 32 use the inbuilt `normcdf` function to generate a CDF based upon the X data values over the range specified in `x`. Care is taken to convert from variance to standard deviation inline to agree with the input to the function.

It should be obvious that some of the code is inefficient. For example, instead of using `norm_p1`, `norm_p2`, etc. the outputs of `normfit` could be placed directly in `param_a` and `param_b`. However, verbosity is favoured for the sake of clarity over terseness for the sake of speed. Were the code to be optimised then these types of changes would need to be applied.

3.6 Confidence

It is important when carrying out work involving these simulations to be confident that any dataset used comes from a simulation which is not, in some unforseen way, anomolous. To provide some degree of condifence, 10 confirmation simulations were carried out, in addition to the simulation which generated the data used in the rest of the thesis. Each run used the 5 node topology with a delay separation of 50ms and a fixed bottleneck at link 1. The delay values for each probe path were collected and the mean and variance taken. The mean of the 10 runs for each probe path was also computed, along with an upper and lower 10% bound. These values are plotted in Figure 3.1 below. The mean delay of each of the ten confirmation simulations is plotted as a cyan dot, the mean delay of the simulation which generated the data used in the thesis is plotted as a blue circle, the mean of the confirmation simulations means' as a red cross and the upper and lower bounds as black plusses.

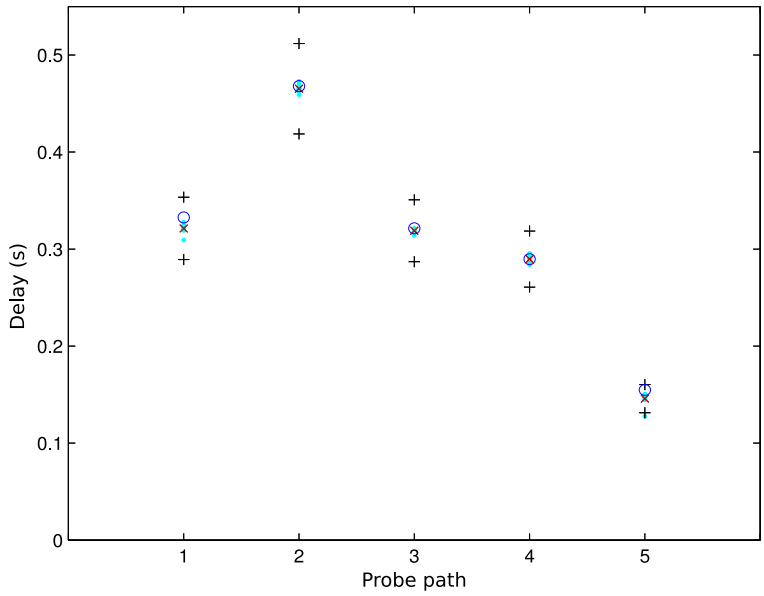


Figure 3.1: *Confidence limits (black), the mean delays of the confirmation data (cyan), the mean of the main data (blue) and the mean of the means of the confirmation data (red) plotted for each probe path for the 5 node topology with 50ms separation.*

What can be observed is that the data used for the main work of this thesis lies within the 10% bounds of the confimation runs giving some assurance that the dataset is not particularly anomolous and the results and conclusions drawn from using it are not misled.

Table 3.1 lists the data used to generate Figure 3.1.

Dataset	Probe 1	Probe 2	Probe 3	Probe 4	Probe 5
Run 1	0.3246	0.4621	0.3176	0.2862	0.1489
Run 2	0.3192	0.4657	0.3190	0.2911	0.1448
Run 3	0.3208	0.4592	0.3142	0.2888	0.1503
Run 4	0.3211	0.4645	0.3194	0.2899	0.1463
Run 5	0.3277	0.4729	0.3216	0.2955	0.1506
Run 6	0.3215	0.4695	0.3203	0.2940	0.1456
Run 7	0.3225	0.4659	0.3193	0.2905	0.1477
Run 8	0.3231	0.4626	0.3179	0.2873	0.1482
Run 9	0.3093	0.4653	0.3190	0.2841	0.1277
Run 10	0.3233	0.4649	0.3208	0.2894	0.1480
Mean of means	0.3213	0.4653	0.3189	0.2897	0.1458
+10% bound	0.3535	0.5118	0.3508	0.3186	0.1604
-10% bound	0.2892	0.4187	0.2870	0.2607	0.1312
Thesis data	0.3327	0.4680	0.3214	0.2897	0.1549

Table 3.1: *Mean values of packet delay for each confimation run on each probe path and other data used to describe the confidence limits.*

Given the nature of traffic and previous discussion (Section 2.4.3) about modelling it, it is also informative to consider the KLD between the data used for the later work and the confirmation runs for the packet delay on each probe path. This is shown in Table 3.2. Since the KLD is empirical, no assumptions need to be made about the distributions of the data whereas the above could be misleading if large outliers were present. The KLD is covered in more detail in Section 6.3.1.

Dataset	Probe 1	Probe 2	Probe 3	Probe 4	Probe 5
Run 1	0.0130	0.0081	0.0044	0.0045	0.0125
Run 2	0.0352	0.0044	0.0038	0.0037	0.0406
Run 3	0.0306	0.0103	0.0122	0.0016	0.0123
Run 4	0.0288	0.0060	0.0049	0.0036	0.0319
Run 5	0.0096	0.0070	0.0033	0.0094	0.0113
Run 6	0.0277	0.0036	0.0033	0.0047	0.0332
Run 7	0.0268	0.0062	0.0052	0.0038	0.0331
Run 8	0.0205	0.0060	0.0040	0.0034	0.0158
Run 9	0.1090	0.0042	0.0045	0.0095	0.3379
Run 10	0.0201	0.0056	0.0035	0.0036	0.0205

Table 3.2: *KLD between the data generated in the confirmation runs and the data used in the rest of the thesis.*

What the numbers in the Table 3.2 shows is that there is a low divergence between the empirical

PDFs of the confirmations runs, and the data used used in the rest of the thesis. The adds to the confidence, gained from the previous results, that the data used in the rest of the thesis does not come from a particularly anomolous simulation.

3.7 Conclusion

The purpose of this chapter was to demonstrate how a network simulation and the subsequent data processing may be carried out when performing work on network tomography. The process was considered to be composed of three steps: network simulation, data extraction and conversion and signal processing. Each step uses a different language and tool and although there are many options available it is believed that the tool-chain chosen is efficient yet flexible. Through example code it was seen that it was necessary to keep file sizes small by generating and retaining only pertinent information at each step to minimise data processing time and storage requirements. Finally, results from a set of confirmation runs were presented. These showed, using confidence bounds and the KLD, that the simulation data used in latter parts of this thesis are not anomolous or outliers.

Chapter 4
Single Gaussian based estimator

Introduction

This chapter explores the use of a single Gaussian distribution (GA) to approximate the packet delay distribution on both a link and a path in a network as part of a network tomography algorithm used to detect bottleneck links. The use of a GA was suggested in [3] but problems with identifiability [82] were raised and no comparison with other models has been carried out. Within the framework of bottleneck link detection [3], the GA is compared with the method of moments (MOM), which was first introduced in [1], with the aims of, determining the feasibility of using a GA in network tomography as a delay estimator, finding the limitations of both estimators, and providing a comparison of their performance in a set of scenarios.

This chapter is divided into four sections. The first section (4.1) reviews some of the statistical properties of Gaussian distributions and examines sampled data from a network which justifies the later assumption that the GA is a suitable model for packet delays. In the second section (4.2) two previously proposed estimation algorithms and their accompanying detection algorithms are discussed. In the third section (4.3), there is a discussion of some of the specific details of the simulations, which follow the same method and use the same tool-chain as described in Chapter 3. Included are the key parameters and the reasons for their choice. The fourth section (4.4) presents a comparison of the two methods using simulation results and makes some observations based on the data gathered.

4.1 A review of some properties of the Gaussian distribution

The Gaussian or normal distribution is a well known and well characterised distribution, and is fully described by its two parameters, the mean, respresented by μ, and the variance, represented by σ^2. The Gaussian distribution has one interesting property which is exploited in this work. It is derived from the central limit theorem and it is that the Gaussian distribution represents the limiting distribution for a sum of statistically independent random variables of

any distribution, as the number of items being summed tends to infinity. This is useful because it can be assumed that for a large data set, where multiple streams of data are being aggregated, each with its own generating distribution, that the overall distribution will tend to a single Gaussian distribution. With any distribution used in this work, the cumulative distribution function (CDF) form is preferred for manipulation and study. The Gaussian CDF is as shown in equation (4.1).

$$f_j(x) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right) \tag{4.1}$$

4.1.1 Justification for the choice of a Gaussian distribution as an estimator of delay

Consider the CDFs of packet delays on some links in a network such as the one shown in Figure 4.1. In this example, the delay on each of the four links has been defined to have a specific minimum value: 10ms, 80ms, 100ms and 150ms respectively. The distributions do not appear to take the form of any standard parametric distributions such as Poisson, log-normal or exponential; therefore using such a distribution as an estimator will always result in an error of fit. Observe that the CDFs are not well separated with links 1, 2, and 3 converging at around $F(x) = 0.72$. A choice must be made as to which distribution will offer the best fit (in terms of least error) and be least difficult to estimate. A complex distribution may offer an excellent fit but at the cost of high computational complexity.

Considering the properties outlined above, a single Gaussian distribution is chosen to approximate the delay distribution. Figure 4.2 shows the packet delay on two paths plotted on a QQ plot; this plots the given data against a normal distribution with quantiles equal to that of the data. If the data is normally distributed then the data points will lie on the dashed line; if it is not they will deviate from it. Probe 2 has a reasonably close fit to the Gaussian with a small deviation at the upper and lower tails which is not unexpected as it was earlier predicted that the CDF was not a standard parametric distribution. Probe 5 deviates significantly at both tails. The lower tail tends to a horizontal line which indicates that there is an abrupt limit to the distribution and that the CDF has a lower tail which is similar to that of Link 4 in Figure 4.1. This is because each path is formed from multiple links which each have have a minimum delay; therefore each path has a minimum delay which appears as a deviation at the lower tail.

It should be noted that a single Gaussian distribution is a fat-tailed distribution which implies that events at the end of the tails of the distribution are more likely than events at the end of the tails of a long tailed distribution. The implication is that using the single Gaussian distribution as a model for the delay distribution will result in over-estimation of the probability of occurrence of events at the ends of the tails. Using a long tailed distribution would result in these *rare* events having a lower probability of occurrence. Additionally, this implies that the mean of the single Gaussian distribution should be treated with caution as it is skewed by *rare* events.

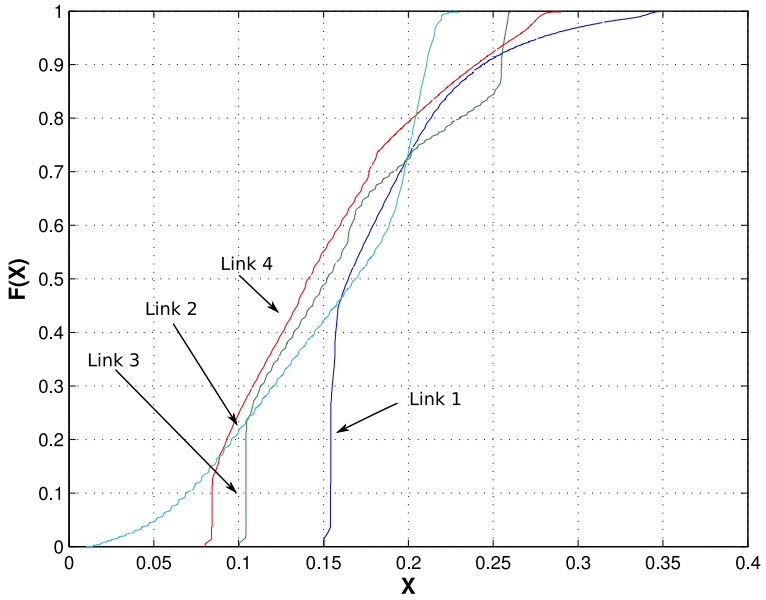


Figure 4.1: *CDFs of packet delay on each links in a 5 node network topology. The difference between the minimum delay of the bottleneck link, link 1, and the next worst link, link 3, is 50ms.*

4.2 Estimation and Detection Procedures

Network tomography algorithms for bottleneck link detection can be separated into two parts: the estimation of the of the link data (the moments in MOM or the CDF in GA), and the detection of the bottleneck link.

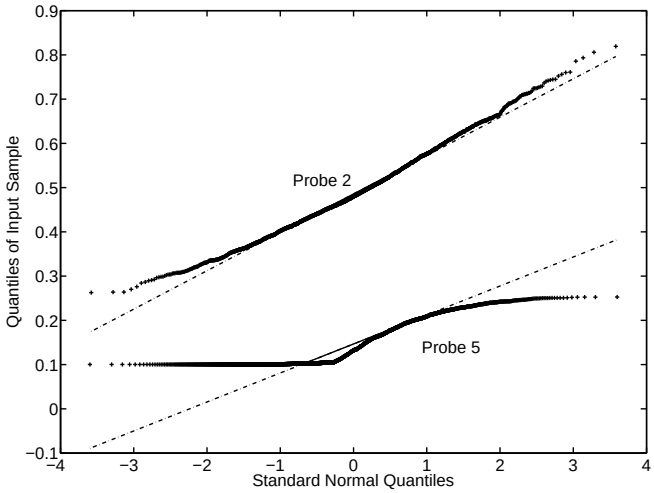


Figure 4.2: *The data from paths 2 and 5 (thick line) of a 5 node network topology plotted against that of a normalised Gaussian (thin line). Path 5 diverges from the Gaussian at the lower tail because of the minimum delay on that path.*

4.2.1 The single Gaussian distribution

It is assumed that the distribution of packet delays on each link and on each path can be modelled by a single Gaussian distribution. Using equation (4.2) for the mean and equation (4.3) for the variance, it is possible to estimate the parameters of path i from N measured delays (Y_{ik} is the k^{th} delay on path i).

$$\widehat{\mu_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} Y_{ik} \tag{4.2}$$

$$\widehat{\sigma^2_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} (Y_{ik} - \widehat{\mu_{Y_i}})^2 \tag{4.3}$$

Using the least squares (LS) algorithm, which was introduced in Section 2.4.1, it is possible to convert path estimates into link estimates; equation (4.4) shows how the delay on link j can be expressed using the estimates gained above in terms of a Gaussian distribution and the elements of the inverse routing matrix where $h_{i,j}$ represents the element in the i^{th} row and j^{th} column of H^{-1}.

$$X_j \sim \mathcal{N}(\sum_{i=1}^{P} \widehat{\mu_{Y_i}}.h_{i,j}, \sum_{i=1}^{P} \widehat{\sigma^2_{Y_i}} \cdot |h_{i,j}|^2) = \mathcal{N}(\widehat{\mu_{X_j}}, \widehat{\sigma^2_{X_j}}) \tag{4.4}$$

The CDF of the delay on link j can be expressed as the standard Gaussian CDF using the previously estimated parameters as shown in equation (4.5). The algorithm is shown in Algorithm 1.

$$f_j(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x - \widehat{\mu_j}}{\widehat{\sigma_j}\sqrt{2}}\right)\right) \tag{4.5}$$

Algorithm 1 GA estimation algorithm

1: **for** $i = 1$ to P **do**
2: fit a single Gaussian distribution to delay values for path i using equations (4.2) & (4.3)
3: **end for**
4: **for** $j = 1$ to L **do**
5: estimate CDF of link j using path estimates and LS as in equation (4.4)
6: **end for**

4.2.2 The method of moments

If no assumptions are made about the distribution of packet delays on a link or a path, it is still possible to estimate the statistics of the path data using the MOM.

Using equation (4.6), the moment generating function (MGF) of path i can be estimated from N measured packet delays. Only the first 20 moments (i.e. $t = 20$) are estimated here unlike [3] as it is expected that the volume of data required to get a reasonable estimate of the higher order moments may exceed the volume of data available.

$$\widehat{M_{Y_i}(t)} = \frac{1}{N}\sum_{k=1}^{N} e^{tY_{ik}} \tag{4.6}$$

Using the LS algorithm the estimated path MGF can be converted into an estimated link cumulant generating function (CGF). Equation (4.7) shows how the CGF estimate for link is j formed. The algorithm is shown in Algorithm 2.

$$\widehat{K_{X_j}} = \sum_{i=1}^{P} h_{ij} \cdot \log(\widehat{M_{Y_i}}) \tag{4.7}$$

Algorithm 2 MOM estimation algorithm

1: **for** $i = 1$ to L **do**
2: estimate MGF of delay values for path i using equation (4.6)
3: **end for**
4: **for** $j = 1$ to L **do**
5: estimate CGF of link j using path estimates and LS as in equation (4.7)
6: **end for**

4.2.3 The Chernoff Bound

The MOM outputs a CGF estimate and therefore to detect the bottleneck link an algorithm is required to compare link CGFs. This is unfortunate as comparison of CDFs is preferred[1] however there is no method to easily convert a CGF to a CDF. Instead, it is possible to use the Chernoff bound as in [1] and shown in equation (4.8). The Chernoff bound returns P_j, the probability of link j exceeding delay threshold δ.

$$P_j = P(X_j \geq \delta) \leq \min_{t>0} \left(e^{-t\delta} \cdot e^{tK_{X_j}} \right) \tag{4.8}$$

The link with the highest value of P_j is selected as the bottleneck link, however, there is a problem in that the value of δ must be selected before computing P_j. If the bound is sensitive to the choice of δ, then the performance of the estimator could be obscured by poor choice of δ. It is expected that in a real scenario a range for δ would be empirically derived. It can be see from 4.8 that the value of the CGF parameter t also has a bearing on the detection algorithm. It was found that it is possible to lower t to 1 without any change of performance however this result may be limited to the datasets used and the range of the delay values contained within them.

[1] CDF comparison is preferred because visually, it is much easier to judge the separation of links and the shape of the CDF curve can give an insight into the spread of the distribution. This comparison could be accomplished using the probability density function (PDF) but the CDF is used as it simplifies the CDFmax algorithm and conversion from PDF to CDF is simple.

4.2.4 CDFmax

Parametric estimation methods such as GA output a CDF estimate and therefore to detect the bottleneck link an algorithm is required to compare CDFs. Were the distributions always Gaussian then the complementary error-function (erfc) could be used but a more general method is sought.

δ is defined to be the delay threshold which is analogous to the delay threshold used in the Chernoff bound. More specifically, it is the value at which all of the CDFs are evaluated. The δ notation is retained because it serves the same purpose in CDFmax as in the Chernoff bound. The CDF of each link is evaluated at δ and the bottleneck link is picked to be the link with lowest value. The link with lowest value at δ has the highest probability of having delay values above the delay threshold and therefore its values of delay are likely to be higher than that of the other links. As with the Chernoff bound, if this algorithm is sensitive to the value of δ then the performance of the estimator may be degraded. This is caused by the detection algorithm being unable to detect the bottleneck link from amongst closely spaced CDFs. Equation (4.9) shows the evaluation of P_j at δ using similar terms and in a similar manner to equation (4.8). The bottleneck link will be detected as the link with the lowest value of P_j.

$$P_j = \mathrm{CDF}_j(\delta),\ j\epsilon\{1, 2, \ldots, L\} \tag{4.9}$$

4.3 Simulation details

The simulations used to evaluate the GA and MOM methods were performed using the tool-chain as described in Chapter 3. Presented in Table 4.1 are the key network and traffic parameters which define the scope and form of the simulations; these parameters are discussed in further detail in subsequent sections.

Simulation time was chosen to be 5000 seconds to give a suitably long data set that could be divided into blocks to perform averaging. In general, a block size of 20 seconds was used, giving 250 blocks for each simulation which allowed the averaging of detection accuracy results to reduce the effect of any outliers. This is a reasonable approach to take given that the data used has been seen to be consistent and with low variability (Section 3.6). The other option would have been to use a shorter length simulation and repeat each experiment in a *Monte*

Parameter	**5-node Network**	**10-node Network**
Added delay on bottleneck link	20ms, 50ms	50ms, 250ms
Added delay on normal links	10ms, 80ms, 100ms	100ms
Link bandwidth on each link	1 Mbps	1 Mbps
Simulation Time	5000 s	5000 s
Number of paths, P	5	12
Number of links, L	4	9
CGF Parameter, t	20	20
Number of probe packets, N	32000	32000
Probe packet rate	2 Kb/s	2 Kb/s
Probe packet size	40 Bytes	40 Bytes
Block size, B	20s	20s

Table 4.1: *A summary of key simulation parameters used throughout this thesis.*

Carlo manner. The former, the approach used here, is less computationally expensive.

4.3.1 Topologies used

In this thesis, two topologies are used: the first comprises 5 nodes, 4 links and 5 paths and the second comprises 10 nodes, 9 links and 12 paths. These are referred to throughout this thesis as the 5 node and 10 node topologies respectively. The 5 node topology shown in Figure 4.3 was chosen to match that in [1] in order to compare results directly. The 10 node topology shown in Figure 4.4 is an extension based upon the 5 node topology and used to determine robustness in a more complex scenario. The 10 node topology was designed to have a routing matrix (RM) that is of full rank, as the 5 node topology does.

4.3.2 Link specifications

Each link in the simulated topology has a set of common, key parameters which are as shown in Table 4.2.

Parameter	**Value**
Directionality	Duplex
Symmetry	Symmetric
Bandwidth	1Mbps
Queue length	50 packets
Queue type	Droptail

Table 4.2: *Common, key link specifications used throughout this thesis.*

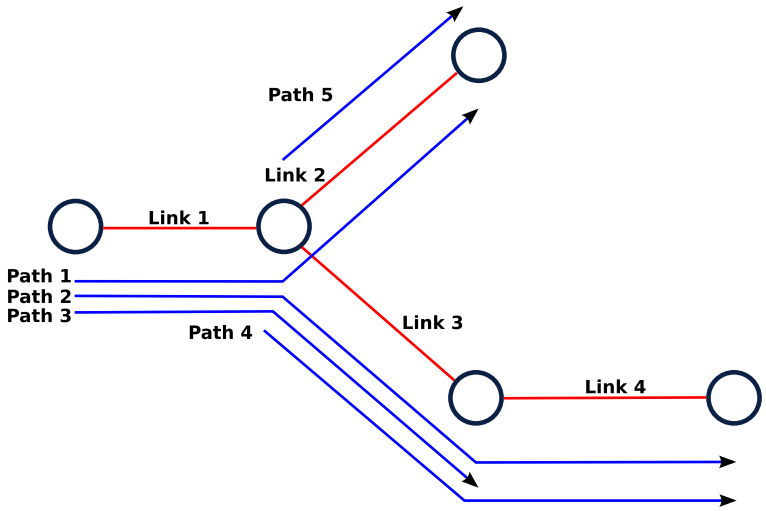


Figure 4.3: *The topology of the 5 node network with 4 links (shown in red) and 5 probe paths (shown in blue). This is based on the topology used in [1].*

In order not to over-simplify traffic flow modelling, bi-directional (duplex) links are used. Traffic can flow in both directions with the forward direction generally the direction of the background and probe traffic, and the reverse direction being that of the acknowledgement and routing traffic. A bandwidth of 1 Mbps is chosen as a feasible but arbitrary value; it could be greater to be more realistic but this is simply a matter of scaling. The queue length is similarly chosen to be sensible but arbitrary and the queue type is that which most simply implements a standard first-in first-out (FIFO) queue.

On each link, a delay was defined such that each packet passing through the link would encounter this delay. These values were selected to give scenarios for the estimator and detector algorithms of varying difficulty. This was necessary because self-congestion on the links due to background traffic alone could not produce a wide enough variety of scenarios for testing.

4.3.3 Network traffic types

There are two types of traffic used in each simulation. The first type is referred to as the background traffic, data packets which flow between two directly connected nodes and used to ensure that links are operating at near capacity so that packets experience congestion and queueing. The second type is referred to as the probe traffic, data packets which follow the paths defined previously and are used to measure the statistical properties of the network. Details

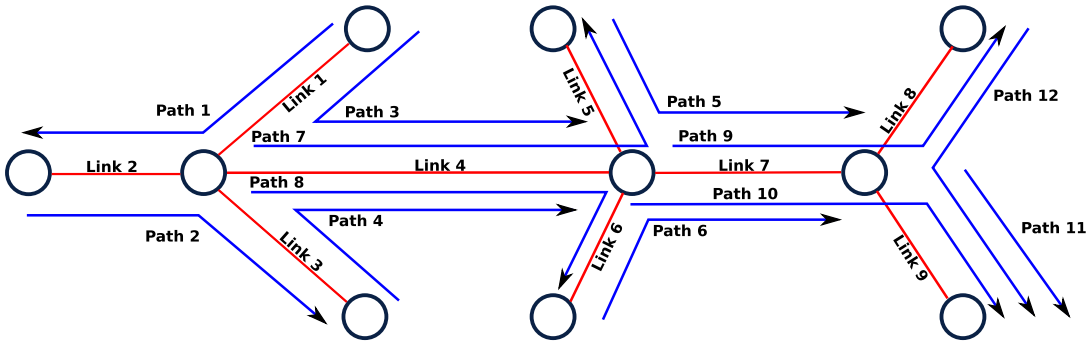


Figure 4.4: *The topology of the 10 node network with 9 links (shown in red) and 12 probe paths (shown in blue). This is an extension of the 5 node topology and designed such that the routing matrix is of full rank.*

about both types of traffic can be found below.

4.3.3.1 Background traffic

The background traffic on each link is chosen to be an aggregation of user datagram protocol (UDP) and transmission control protocol (TCP) packets respresenting a mix similar to that of measured network links (from sources such as [49]).

Parameter	**Value**
Packet generating application	Pareto
Shape	1.5
Burst time	500 ms
Idle time	500 ms
Packet size	512 Bytes

Table 4.3: *UDP agent specifications for generating background traffic.*

Parameter	**Value**
Packet generating application	FTP
Packet size	512 Bytes
Initial window size	1460 Bytes
Upper window size limit	46720 Bytes

Table 4.4: *TCP agent specifications for generating background traffic.*

The fixed load part of the traffic mix is defined using a number of UDP agents running a Pareto application to generate a base load of approximately 80 to 90% of the link bandwidth in order to ensure self congestion on each link. The number of agents on each link is in the range 1 to 5; the bandwidth of each link is 1 Mb and therefore the rate of the Pareto application is set to between 200kb/s and 800kb/s. For these agents the parameters are as shown in Table 4.3.

The variable load part of the traffic mix is defined using a number of TCP agents running a file transfer protocol (FTP) application, for which a rate cannot be set, in order to ensure that each link will be heavily loaded since TCP will adjust its data rate to match that of the available bandwidth. There are between 1 and 3 agents on each link. The FTP application generates packets with a size of 512 Bytes (inclusive of TCP/internet protocol (IP) headers) and has an initial TCP window size of 1460 Bytes (1 packet). Both these values are adjusted by TCP as it segments packets to ensure most efficient use of available bandwith and also adjusts both send and receive window sizes for the same reason. The specifications for these agents are as shown in Table 4.4.

4.3.3.2 Probe traffic

Probe traffic is generated using a UDP agent running a constant bit-rate (CBR) application which outputs unicast, 40 Byte packets, at a rate of 2 Kb/s. To reduce the correlation between probe packets which may arise from each agent starting at the same time, there is a 1 second delay between agent start times. As all packet streams run for a fixed length of time, they are therefore staggered with respect to each other in time. It is assumed that the network statistics are stationary or close enough to stationary so that this does not affect our algorithm performance.

The probe rate and packet size are comparable with those in [1]; they appear sensible choices and are used here in order to directly compare results with those published by Coates, Nowak et al.

4.4 A comparison of the performance of GA and MOM

A method is defined as the combination of an estimation algorithm and a suitable detection algorithm. In this thesis the nomenclature adopted is that the method is referred to by the

name or acronym of the estimation algorithm. The GA method is the combination of the GA estimation algorithm and the CDFmax detection algorithm and the MOM method is the combination of the MOM estimation algorithm and the Chernoff bound detection algorithm.

The GA method and the MOM method are compared using the three metrics as given below. Six different simulation scenarios are used to perform the comparison. The first scenario uses the 5 node topology (Figure 4.3), with 3 values of separation: 20ms, 50ms and 150ms. The second scenario uses the larger, 10 node topology (Figure 4.4), with separations of 50ms, 100ms and 250ms.

The performance metrics are:

- bottleneck link detection accuracy - how often will the method correctly identify the bottleneck link?
- computational complexity - how many operations are necessary to perform estimation and detection?
- robustness - how does the detection accuracy change with changing network size and separation value?

It was necessary to validate the operation of the MOM method as all subsequent methods are compared against it, using the metrics noted above. The validation was performed by plotting the cumulants in the range $[1, 200]$ and comparing with Figure 6 of [1]. The closeness of fit observed, by considering both shape and scale, established that the implementation of the MOM method used in this work was valid and suitable as a representation of the method introduced in the literature.

4.4.1 Bottleneck Link Detection Accuracy

In this section, the accuracy of bottleneck link detection is plotted against the value of the delay threshold δ. The range of δ is chosen to span the range of values in which the CDFs are expected to lie, [0.1s, 0.25s]. When describing the various scenarios, the separation is defined as the difference in delay between the bottleneck link and the next worst link. For example, if the delay on the bottleneck link is 120ms and the delay on the next worst link is 100ms then the separation will be quoted as 20ms.

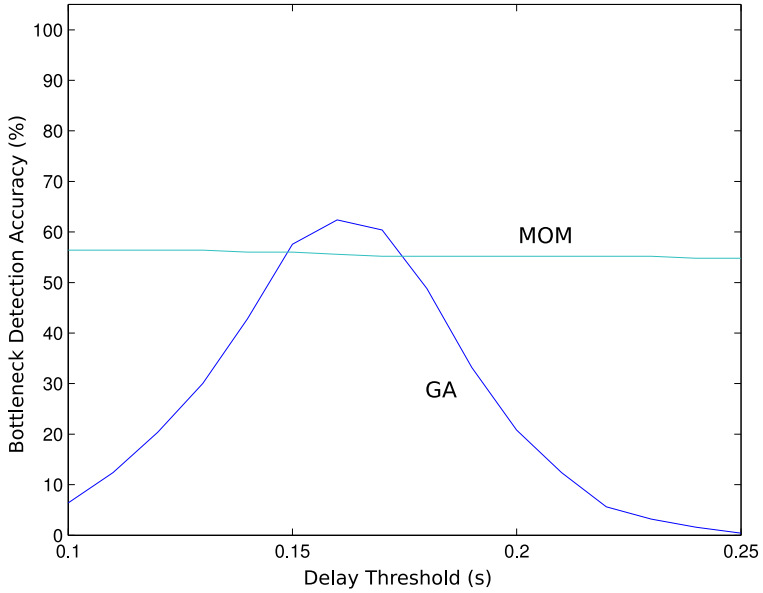


Figure 4.5: *Bottleneck link detection accuracy results for GA and MOM for the 5 node topology with 20 ms separation.*

Figure 4.5 shows the 5 node topology with a separation of 20ms. This shows the main difference between MOM and GA: MOM has a near constant performance, here of approximately 56% across all values of δ, whilst GA is sensitive to δ. GA is more accurate than MOM with a good choice of δ with a peak accuracy approximately 8% higher than with MOM. However, a poor choice of δ results in very poor detection accuracy.

Figure 4.6 shows the 5 node topology with an increased separation of 50ms. Both methods perform with increased accuracy due to the increase in separation. MOM now achieves 96% accuracy and remains insensitive to the choice of δ. GA also achieves 96% accuracy at its peak and does so over a wider range compared to Figure 4.5. Whilst the accuracy for GA does not drop below 50% over the plotted range of δ, the trend suggests that this would roll off to zero.

To gain some insight into why GA is sensitive to the value of δ consider Figure 4.7 where the estimated link CDFs for the 5 node topology with 50ms separation are shown. The CDFs are plotted only over the normal range of δ to enable a clearer view. It is clear that the bottleneck link, Link 1, has in general a lower value than the others which indicates that it is the bottleneck. Note that the curves cross at various points and that at an arbitrary value of δ the bottleneck

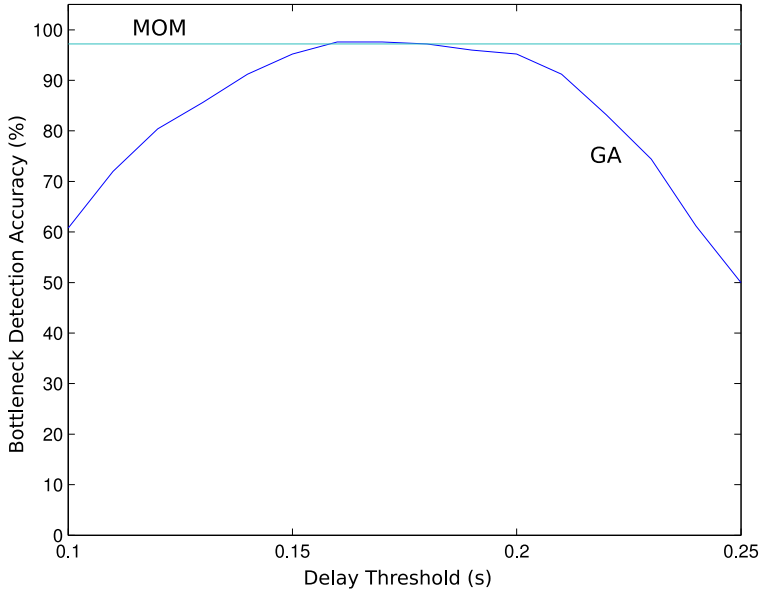


Figure 4.6: *Bottleneck link detection accuracy results for GA and MOM for the 5 node topology with 50 ms separation.*

link may not always have the smallest value of CDF. The CDFs also tend to converge at the upper and lower values of the threshold which gives a reason why GA appears to be sensitive to choice of threshold at the endpoints of the range - there is not enough separation to enable a robust selection of the bottleneck link.

The final scenario for the 5 node topology is that with a large separation of 150ms. Both methods achieve 100% accuracy which is expected with 150ms separation. The bottleneck link CDF is well separated from that of the other links and is therefore easier to detect, even with poor estimation.

From these three figures, it can be seen that GA is a suitable method for this scenario and that it can achieve higher detection accuracy than MOM. With a low separation between the bottleneck and the next worst links it is sensitive to δ which is not the case with MOM. Although it does show a small amount of variation, it is not of the same scale as that of GA, at worst, the variation is roughly 2% over the range of δ.

Figure 4.8 shows the larger, 10 node topology with a separation of 50ms which is the most challenging scenario for any method. MOM accuracy is zero at all values of δ which is consistent with the previous results. This may indicate the presence of a minimum separation

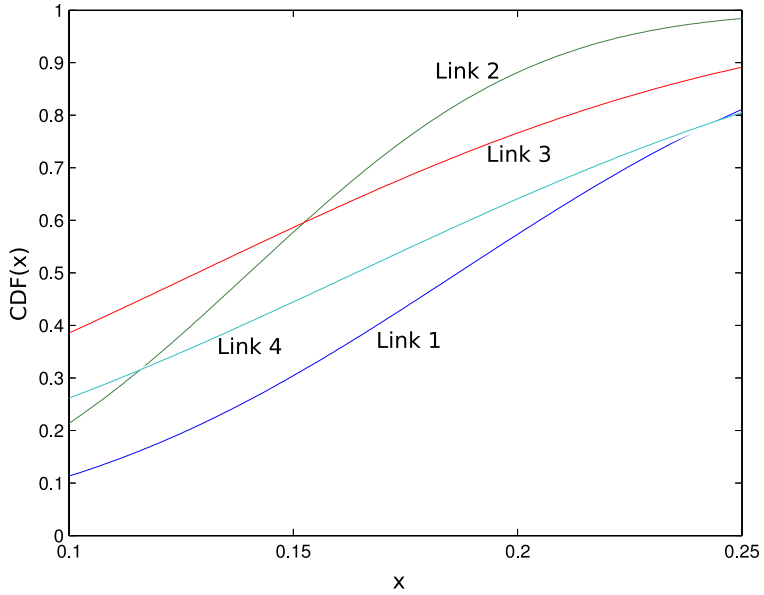


Figure 4.7: *Example link CDFs estimated using the GA estimation algorithm for the 5 node topology with 50ms separation.*

required for MOM to operate. GA achieves a peak accuracy of 17% but has a steep roll off on either side. Given the reasonably high accuracy of both estimators in the 5 node topology with the same separation, it was expected that the accuracy here would be similar but it appears that moving to a larger topology has reduced the effectiveness of both methods.

Figure 4.9 shows the 10 node topology with 100ms separation and displays an unusual result for both estimators: MOM accuracy is constant at just under 30% whilst GA achives a peak accuracy of 100% with a large range, greater than that of Figure 4.6; the roll-off is comparable to that seen in Figure 4.6 at the higher end of the range. The low accuracy of MOM may provide more evidence that there is a minimum separation necessary for the method to perform well.

The final scenario for the 10 node topology is one with 250ms separation and it is found that the results are identical to those of the 5 node topology with 150ms, namely, 100% accuracy for both methods. This is interesting because it illustrates that 100% accuracy can be achieved in the 10 node topology, but that it requires a larger separation than with the 5 node topology.

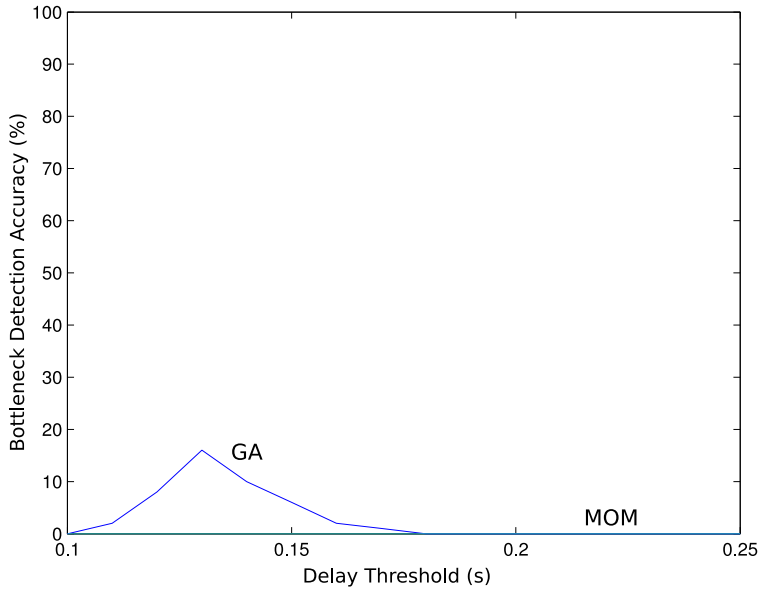


Figure 4.8: *Bottleneck link detection accuracy results for GA and MOM for the 10 node topology with 50 ms separation.*

4.4.2 Computational complexity

Two methods are used to compare the computational complexity required by both methods. The first is to derive a formula in terms of the key parameters for the number of add (ADD) and multiply (MULT) operations required to perform estimation and detection on one block of N data. The second is to measure the mean time required. The complexity formulae for both methods are derived from the code of the MATLAB implementations the algorithms and are shown in Table 4.5.

Method	**MULT**	**ADD**
GA	$L(87+3P)+P(N+2)$	$30L+L(P-1)+3PN$
MOM	$Pt(2N+1)+L(5t+2)$	$PtN+LP+(L-1)!$

Table 4.5: *Complexity formulae for the number of operations required for both methods in terms of the number of samples (N), the number of links (L), the number of paths (P) and the CGF parameter (t).*

It can be seen that the complexity of both methods scales with the number of data samples, N, and the size of the network, P and L. Additionally, MOM scales with the CGF parameter t, the number of cumulants estimated. The accuracy of the MOM estimation algorithm is therefore

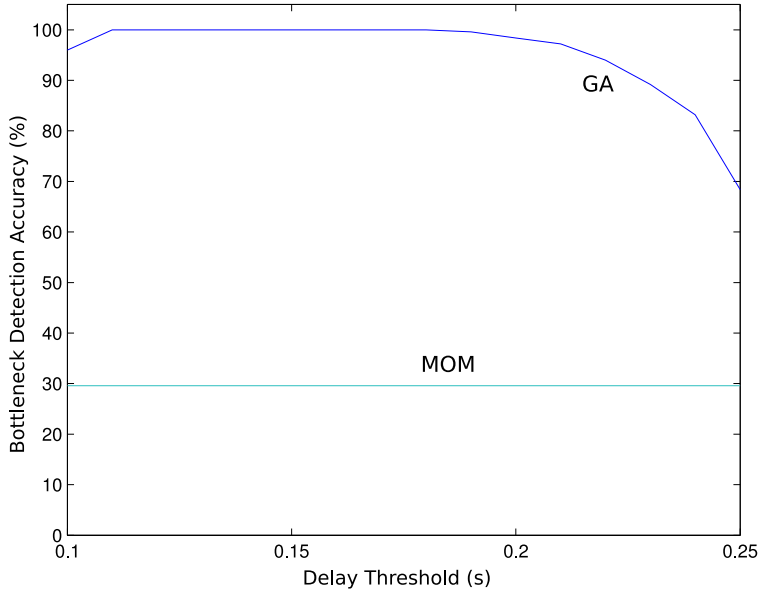


Figure 4.9: *Bottleneck link detection accuracy results for GA and MOM for the 10 node topology with 100 ms separation.*

dependant upon it. It was previously noted that empirical adjustment can be used to reduce the complexity and in the scenarios used in this thesis $t = 1$ will work. In order to retain flexibility, to cope with variable data ranges which may require a larger value of t, the value is fixed at 20. Table 4.6 shows the number of operations required by both methods for the 5-node network in processing one 20 second block of data. N is calculated as $(B \times 2\ \mathrm{Kb/s})/40\ \mathrm{Bytes} = 128$ where B is the block size in seconds, 2 Kb/s the probe traffic data rate and 40 Bytes the probe traffic packet size.

Method	**MULT**	**ADD**
GA	1058	2056
MOM	26,108	12,826

Table 4.6: *The number of ADDs and MULTs required by both methods to perform estimation and detection on one block of N delay samples.*

GA is approximately an order of magnitude less complex than MOM in both ADDs and MULTs. It is probable that there is a connection between this and the fact that GA estimates an order of magnitude fewer parameters than MOM.

4.4.3 Robustness

Another useful metric to know is how both methods behave as the volume of data available to them decreases. To view this, consider the accuracy plotted against the probe packet rate as shown in Figure 4.10.

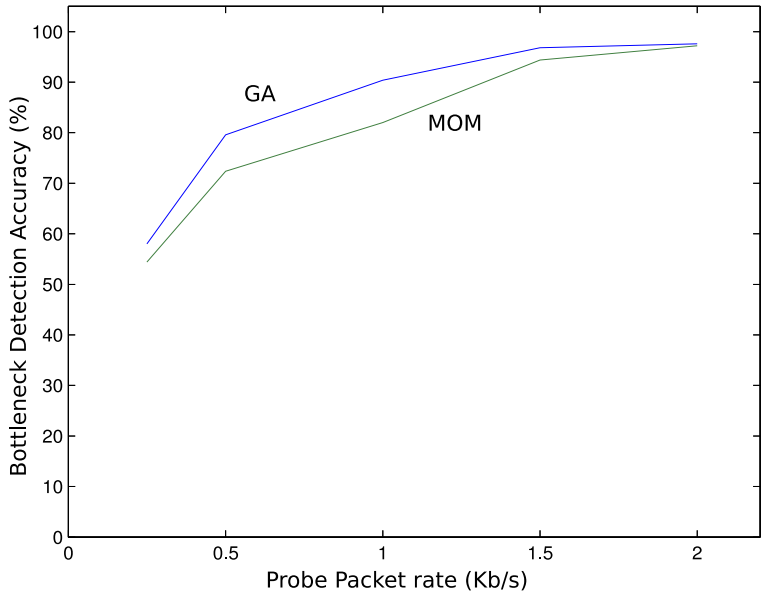


Figure 4.10: *Bottleneck link detection accuracy results for various estimator rates for both MOM and GA using the 5 node topology, 50ms separation and* $\delta = 0.16$.

Using the 5 node topology, a separation of 50ms and $\delta = 0.16$, the value at which the best accuracy was shown in Figure 4.6, the results of reducing the number of probe packets in the network from 2 Kb/s to 0.25 Kb/s can be observed. The result for 2 Kb/s is identical to that of Figure 4.6, as expected, and it is seen that GA always has higher accuracy than MOM although often only by a small margin. This could be due to MOM estimating higher order moments and therefore requiring a larger number of delay samples and that the value of δ was equal to that where the peak accuracy occurred. It also appears that both methods are more robust to this reduction in data than they are to lowering separation.

Figures 4.6 and 4.5 show a degradation in MOM accuracy from just under 100% to just over 55% as the separation decreases. Figure 4.11 shows some additional results using the 5 node topology, with $\delta = 0.15$, which suggest that accuracy is not linearly related to separation. It can be seen that to get 100% accuracy, a large separation is required (150ms) whilst to get close to this (98%), only 60ms is required. This suggests that if an accuracy of less than 100% were

acceptable then MOM can be competitive with GA.

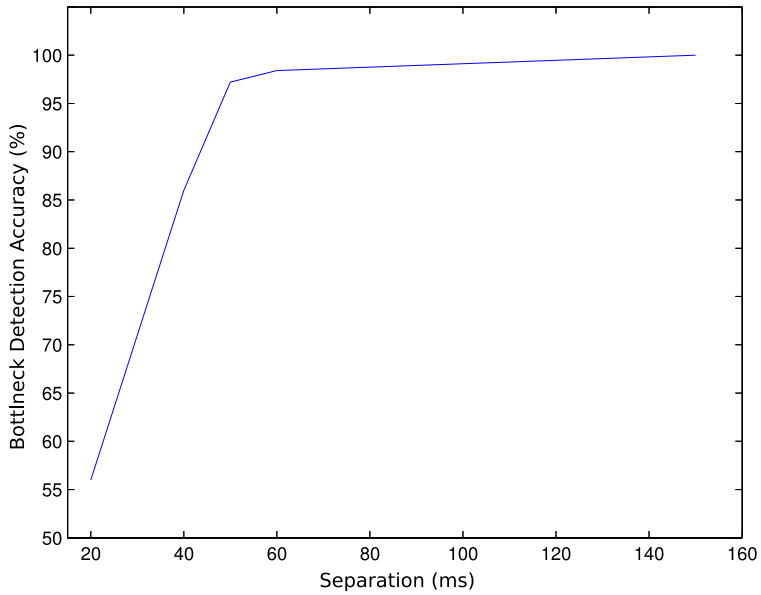


Figure 4.11: *Bottleneck link detection accuracy for MOM for the 5 node topology with separation values of 20, 40, 50, 60 and 150ms with* $\delta = 0.15$.

4.5 Conclusion

In this chapter two estimation and two detection algorithms were introduced. Two methods were formed from appropriate combinations of estimation and detection algorithms and their performance in six scenarios was compared. Based upon bottleneck link detection accuracy it could be concluded that MOM is preferable because it has an accuracy which is robust to changes in delay threshold so long as the separation is greater than a minimum value. Based upon computational complexity and robustness, GA appears preferable because it is less complex and still performs in the most challenging scenario with careful choice of threshold. By considering all metrics a more complete picture appears: if computational complexity is the largest constraint then GA is preferable but it necessitates choosing the delay threshold carefully; if an optimum choice of threshold is not available or impractical then MOM is preferable but it comes with the cost of higher delay separation - especially in a larger network.

Chapter 5
Gaussian mixture models

Introduction

This chapter explores the idea of extending the GA algorithm to make use of a more flexible distribution, namely a univariate Gaussian mixture model (GMM) distribution. The use of mixture models is introduced in [60] where the authors use a hybrid discrete/continuous mixture model and in [5] where the authors use an exponential mixture model. Three new estimation algorithms are introduced: the raw transform (RT) which is used as a lower bound for bottleneck link detection accuracy, the Sum-of-Gaussians B (SOGB), which is a GMM-based extension of RT and the Sum-of-Gaussians A (SOGA) in which the single Gaussian distribution in GA is replaced with a GMM. Once paired with suitable detection algorithms, the performance of the new methods is shown alongside that of GA and MOM using the same scenarios and metrics introduced in the previous chapter.

This chapter is divided into four sections. The first section (5.1) examines the key properties of a GMM and comments on those which are most of use in this work. The second section (5.2) introduces the new methods using the estimation algorithms described above and the existing detection algorithms. The third section (5.3) provides the key simulation parameters necessary to recreate the results which are shown and discussed in the final section (5.4).

5.1 A review of key properties of univariate GMMs

A univariate GMM[1] is a probability distribution which is formed from the addition of two or more univariate Gaussian distributions, which are referred to from here onwards as elements. Each element is fully described by its mean, variance and weight; the mean and variance are as for any Gaussian, the weight indicates how significant this element is to the GMM. An example of a GMM PDF with three elements defined as $\mathcal{N}(-3,1)$ weight=0.3, $\mathcal{N}(0,1)$ weight=0.3 and

[1] A GMM is described as a sum-of-Gaussians distribution in some papers although they are identical concepts. In this thesis, the mixture model nomenclature is used.

$\mathcal{N}(2, 0.25)$ weight=0.4 is shown in Figure 5.1. The advantage of using a GMM is that with a sufficient number of elements, a GMM is flexible enough to model an arbitrary distribution. The choice of the number of elements to use must be made *a-priori* - too few elements will result in underfitting where the approximation to the given distribution will be poor, too many elements will result in overfitting where either elements with small weights or nearly identical elements may occur. The fitting of the elements is done via the expectation maximization (EM) algorithm [53], an iterative, successive approximation algorithm.

The disadvantage of using a GMM is that it is not possible to determine which element should be associated with which link when a GMM is fitted to path data; this is where LS must be used. Given that the exponential mixture model algorithm for delay estimation was introduced in [5] there is some justification for using mixture models. The choice of the GMM is based on the desire to extend the GA method which had shown promising performance in the previous chapter.

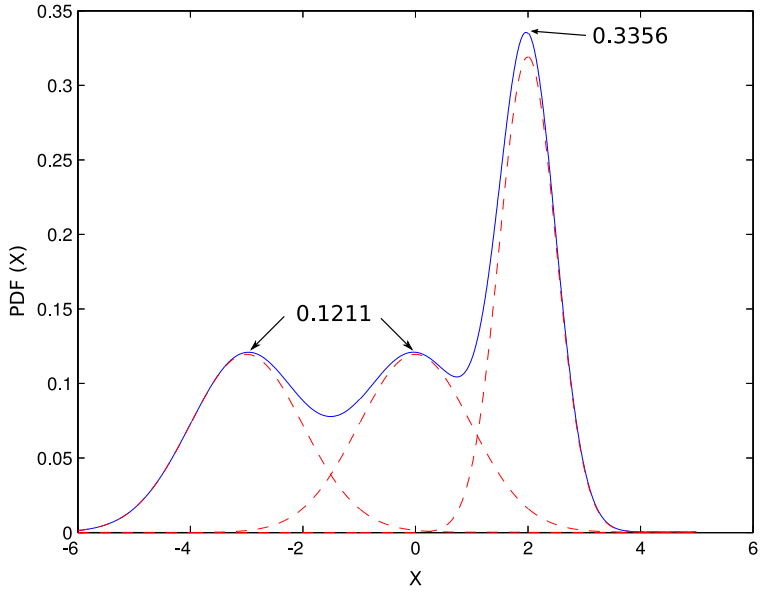


Figure 5.1: *An example of a GMM PDF with three elements - $\mathcal{N}(-3, 1)$ weight=0.3, $\mathcal{N}(0, 1)$ weight=0.3 and $\mathcal{N}(2, 0.25)$ weight=0.4. The GMM PDF is shown in blue, the individual element PDFs are in red.*

5.1.1 Convergence

Unfortunately, the iterative nature of the EM algorithm can lead to large computation times when attempting to fit a GMM to empirical data. To restrain the maximum computation time, the EM algorithm is restricted using both a convergence limit and a maximum number of iterations. The algorithm is stopped when either limit is reached. The iteration limit, the maximum number of iterations the algorithm may perform, is specified in advance. When the limit is reached, the values for that iteration are taken as the result. The second limit is a convergence limit and is also specified in advance. Because the EM algorithm tries to fit a mixture model to the data, it is possible that it will never find a mixture with a *perfect* fit. The convergence limit is a value which is compared to the output of successive iterations. The algorithm is stopped when the difference between successive iterations is lower than the limit. With both limits, it is assumed that the algorithm will converge asymptotically to a solution. The algorithm is not gauranteed to converge for an arbitrary dataset, or to converge to the global optimal. It may diverge from the starting value, away from the global optimum. In this case the results may not be usable.

In this thesis both limits were chosen empirically. This was done after observation of the algorithm being applied to the datasets and each limit adjusted to allow the algorithm to converge upon a usable set of parameters. The starting values were similarly chosen. The means were the 10^{th}, 50^{th} and 90^{th} percentiles of the data, the variances that of the data and the weights $1/3$.

5.2 Estimation and detection procedures

In this section three new estimation algorithms are introduced: SOGA and SOGB are GMM based extensions of other algorithms while RT provides a lower bound on performance. The detection algorithms which were introduced in the previous chapter can be directly coupled to the estimation algorithms introduced here.

Whilst the GMM nomenclature is used throughout this work, the naming of two of the algorithms (SOGA & SOGB) is based upon the alternate sum-of-Gaussians nomenclature. Since SOGA & SOGB were used in a published paper [83], that naming is retained here for comparison. In addition, the SOGA algorithm was the first to be worked on hence the A designation. However, it is easier to illustrate the use of a GMM by comparison to an existing

algorithm (RT) therefore SOGB is introduced before SOGA.

5.2.1 The raw transform

The RT is, conceptually, the simplest algorithm for estimating the distribution of delays on links and, as such, can be used as a benchmark against which to compare other algorithms. The operation of RT is illustrated in Figure 5.2, described formally in Algorithm 3 and operates as follows.

First, the path delay samples $Y_1 \dots Y_P$ are constructed by taking the mean of the packet delays on each path over the size of the block. In this work, a block size of 1 second is used. For a block of size B_t, the estimated mean of path i in any block is given by (5.1) where Y_{ik} is the k^{th} delay on path i. Secondly, the path samples are transformed, using LS, into a set of link estimates, $X_1 \dots X_L$. Using the nomenclature from the GA method (Section 4.2.1) in which the element of H^{-1} corresponding to path i and link j is $h_{i,j}$, the estimate for link j, X_j, can be given by (5.2). Finally, a histogram is constructed for each link which approximates the PDF of the packet delays on said link.

$$\widehat{Y_i} = \frac{1}{B_t} \sum_{k=1}^{B_t} Y_{ik} \tag{5.1}$$

$$\widehat{X_j} = h_{i,j} \cdot \sum_{i=1}^{P} Y_i \tag{5.2}$$

The RT performs a low volume of manipulation on the data, in comparison to other methods. There is no assumption of an underlying statistical distribution for the delays on any path, or link, akin to the MOM. The manipulation performed is simple averaging - the construction of the blocks - and the transformation from path to link estimates using the LS algorithm. If the assumptions that the delays on each link are independant and that the LS algorithm can transform between path and link estimates without error are true, then the RT should give the most accurate representation of the link data possible and therefore the highest bottleneck link detection accuracy, assuming a suitable detection algoirhtm is used. The performance of RT therefore relies more heavily on the inverse routing matrix than other methods, and is consequently more susceptible to errors in the inverse than other methods. Possible problems

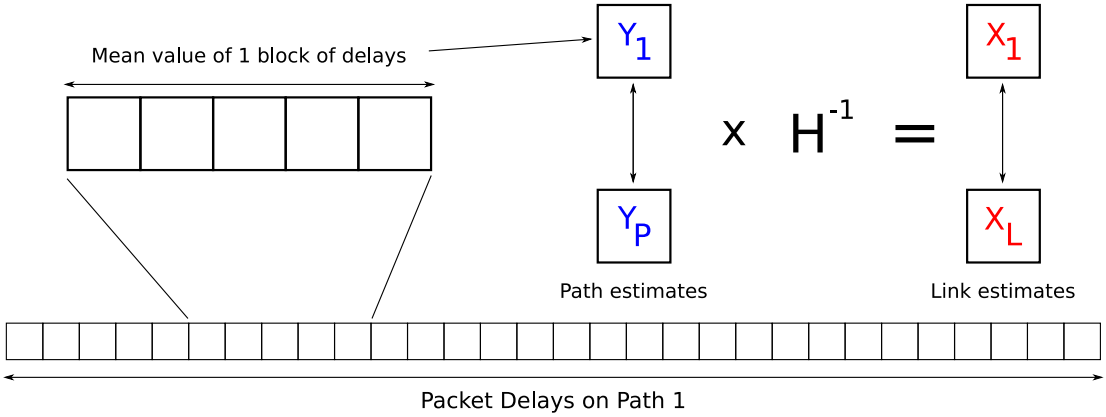


Figure 5.2: *An illustration of the RT algorithm. The path 1 estimate, Y_1, is the mean of the delay over one block. The block shown is 5 delay values in length thus B_t is 5. The link estimates $X_1 \ldots X_L$ come from the transformation by LS of the set of path estimates $Y_1 \ldots Y_P$.*

with the LS algorithm, related to the use of the pseudo-inverse, were discussed in Section 2.4.1.

5.2.2 The Sum-of-Gaussians B

The histogram used in RT is at best a crude method to approximate the PDF of the link packet delays and may require a large volume of data to generate a good estimate. In the Sum-of-Gaussians B (SOGB), the first stage of the RT, the generation of the link samples is used. Instead of the histogram estimation, a three element (i.e. $J = 3$) GMM is is fitted to the link estimates using the EM algorithm. The generation of the link estimates X_j from the path delay samples is as given in (5.1) and (5.2). $J = 3$ is chosen to give some flexibility to the GMM but also to keep computational complexity reasonably low. One disadvantage of SOGB over RT is that fitting the GMM requires multiple blocks of data whereas RT could add a histogram estimate after each block was processed; in this thesis both the histogram and GMM are given the same number of blocks of data to allow like-for-like comparisons. A procedural description of the SOGB algorithm is given in Algorithm 4.

5.2.3 The Sum-of-Gaussians A

In a similar manner to the way in which the RT was extended to form the SOGB, it is possible to extend the GA method, in which it was assumed that the delay on each path could be modelled

Algorithm 3 RT estimation algorithm

1: **for** $B = 1$ to $number\_of\_blocks$ **do**
2: **for** $i = 1$ to P **do**
3: compute the path delay sample Y_i as the mean of the packet delays on link i in this block using equation (5.1).
4: **end for**
5: **end for**
6: **for** $B = 1$ to $number\_of\_blocks$ **do**
7: **for** $j = 1$ to L **do**
8: compute link estimate X_j by converting path samples $Y_1 \dots Y_P$ for block B using LS as in equation (5.2).
9: add link estimate X_j to histogram for link j.
10: **end for**
11: **end for**
12: use link histograms as approximations for link PDFs.

Algorithm 4 SOGB estimation algorithm

1: **for** $B = 1$ to $number\_of\_blocks$ **do**
2: **for** $i = 1$ to P **do**
3: compute the path delay sample Y_i as the mean of the packet delays on link i in this block using equation (5.1).
4: **end for**
5: **end for**
6: **for** $B = 1$ to $number\_of\_blocks$ **do**
7: **for** $j = 1$ to L **do**
8: compute link estimate X_j by converting path samples $Y_1 \dots Y_P$ for block B using LS as in equation (5.2).
9: **end for**
10: **end for**
11: fit a GMM of size J to each link estimate to generate a CDF for each link.

by a single univariate Gaussian to form the Sum-of-Gaussians A (SOGA), where the path delay is modelled using a GMM.

If it is assumed each link is modelled by a single Gaussian distribution, then it can be assumed that any path which is composed of ≥ 2 connected links will have a distribution which is a GMM. As the number of links which compose the path increases, the central limit theorem predicts that the delay distribution will tend to that of a single Gaussian distribution. Here, it is assumed that the number of links which comprise any path is low enough that the assumption about modelling paths using a GMM holds. In this case, the link distribution may be estimated by fitting a GMM to the path data, computing the contribution from each combination of elements, and transforming using the LS algorithm.

This algorithm (shown using the normal algorithmic notation in Algorithm 5) can be best described by the following steps:

1. Fit a GMM of J elements to each path using the EM algorithm.
2. Create a combination table which shows on each row, which of the J elements will represent each of the P paths and the joint weight of all the elements for this combination.
3. Rank the combinations (i.e. sort the rows of the table) in descending order of joint weight. The number of rows in the table may be limited to the T heaviest if desired.[2]
4. Generate a GMM for each link which has a contribution from each combination (i.e. each row) where the mean and variance have been transformed using LS and the weight is that of the joint weight of the combination.
5. If necessary normalise the link estimate to ensure a valid CDF. This is necessary if the number of combinations has been limited to T.

Algorithm 5 SOGA estimation algorithm

```
for i = 1 to P do
    fit GMM of size J to each path, i
end for
for Q = 1 to T do
    compute table of combinations where each path is represented by one element
    compute joint weight for each combination
end for
if constained such that T < J^P then
    normalise weight metrics to sum to 1
end if
for i = 1 to L do
    for Q = 1 to T do
        compute contribution to L from combination Q using LS as in GA
    end for
end for
```

As an example, assume that there is a network in which there are two paths, each fitted with a GMM of three elements which have parameters as shown in Table 5.1. The combination table would be as shown in Table 5.2 where the joint weight is the product of the weights of each

[2] One reason for limiting to T rows is to reduce the complexity in a large network where the possible number of rows increases quickly with the number of links.

choice of element in that combination. For example, combination 7 uses element 3 (weight 0.12) to represent path 1 and element 1 (weight 0.44) to represent path 2. Therefore the joint weight for this combination is $0.12 \times 0.44 = 0.0528$.

Path	Element	Mean	Variance	Weight
1	1	$\mu_{1,1}$	$\sigma^2_{1,1}$	0.70
1	2	$\mu_{1,2}$	$\sigma^2_{1,2}$	0.18
1	3	$\mu_{1,3}$	$\sigma^2_{1,3}$	0.12
2	1	$\mu_{2,1}$	$\sigma^2_{2,1}$	0.44
2	2	$\mu_{2,2}$	$\sigma^2_{2,2}$	0.18
2	3	$\mu_{2,3}$	$\sigma^2_{2,3}$	0.38

Table 5.1: *The parameters used in the SOGA example in Table 5.2.*

Combination	Path 1 Element	Path 2 Element	Joint Weight
1	1	1	0.3080
2	1	2	0.1260
3	1	3	0.2660
4	2	1	0.0792
5	2	2	0.0324
6	2	3	0.0684
7	3	1	0.0528
8	3	2	0.0216
9	3	3	0.0456
		Total	1.0000

Table 5.2: *An example of a SOGA combination table showing the element choice and the joint weight for each combination (row).*

The link estimates in this example would have $J^P = 3^2 = 9$ contributions, one corresponding to each row in Table 5.2, with the path parameters being transformed to link parameter estimates with LS.

The combinatorial step, adding the contributions to each link estimate from each combination, is necessary because the GMM estimates J elements (3 in this work), but does not indicate which link each element should be associated with. Note that LS can transform only one element for each path at a time, and that it provides a method to assign a contribution to each link from each path. It is assumed that the bottleneck link dominates any path which it is a part of, and that the weight associated with the bottleneck element will be high in comparison to the other elements. This would lead to a heavier weight in the combination table, and a

more prominent contribution to the link estimate. Therefore, the GMM implicitly clusters the estimates from each contribution.

Were the process assumed to be ideal, i.e., the LS transform introduced no error, and each link was faithfully modelled by a single Gaussian distribution, then it could be expected that SOGA would generate a number of identical Gaussian distributions for each link. In effect, the algorithm would be clustering *identical* estimates for each link into one estimate.

5.2.4 Detection algorithms

In Section 4.2 CDFmax and the Chernoff bound were introduced as detection algorithms suitable for use with GA and MOM respectively. Like GA, SOGA, RT and SOGB have an output which is compatible with CDFmax so it can be used in the form in which is was introduced. It must be noted that whilst RT generates a histogram estimate of a PDF, transforming a PDF into a CDF requires only basic manipulation.

5.3 Simulation details

The scenarios used in this chapter are identical to those used in the previous one: two network topologies with three values of delay separation for each topology. This allows a comparison of the methods introduced here with the results for GA and MOM. The methods are defined here in the manner they were for GA and MOM: the RT, SOGB and SOGA methods are the combinations of, respectively, the RT, SOGB and SOGA estimators with the CDFmax detector. Key parameters which relate specifically to the methods introduced in this chapter are as shown in Table 5.3, the key parameters given in Table 4.1 remain unchanged. The values of the EM convergence and iteration limits were deduced by empirical observations. The algorithms were run with initial limits which were then altered to accommodate cases where convergence did not take place.

5.4 Results

The results shown below use the bottleneck-link detection accuracy, computational complexity and robustness metrics introduced in Section 4.4 to allow direct comparison between all

Algorithm	Parameter	Value
SOGA	Rows in combination table, T	J^P
SOGA, SOGB	EM convergence limit	0.01
SOGA, SOGB	EM iteration limit	1000 cycles
RT, SOGB	Block size	1 second
RT	Number of blocks per histogram	20
SOGB	Number of blocks per GMM	20
SOGA, SOGB	Number of elements per GMM (J)	3

Table 5.3: *Algorithm specific parameters for RT, SOGB and SOGA used in this thesis. These parameters were chosen based on empirical observations.*

methods. When describing the various scenarios, the separation is defined as the difference in delay between the bottleneck link and the next worst link. For example, if the delay on the bottleneck link is 120ms and the delay on the next worst link is 100ms then the separation will be quoted as 20ms.

5.4.1 Bottleneck link detection accuracy

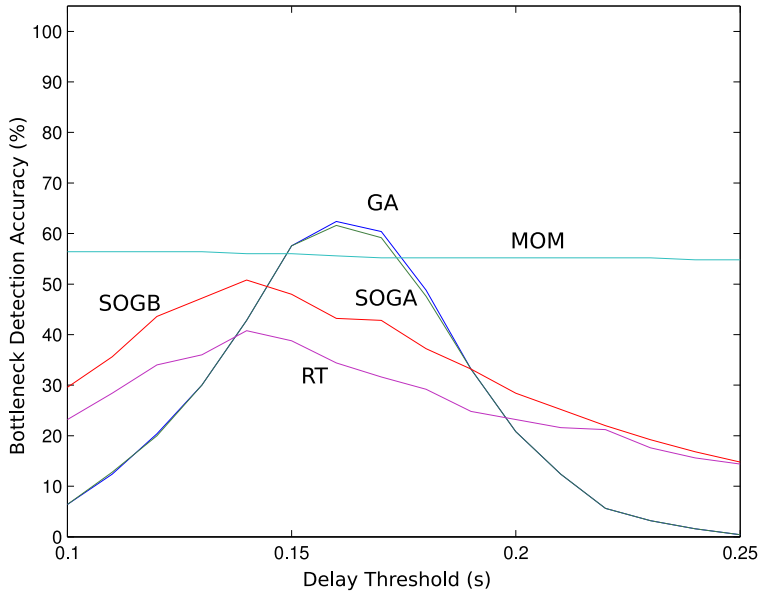


Figure 5.3: *Bottleneck link detection accuracy results for GA, MOM, RT, SOGA and SOGB for the 5 node topology with 20 ms separation.*

Figure 5.3 shows the 5 node topology with 20ms separation where the bottleneck link detection accuracy of all five methods is plotted against the value of the delay threshold δ. RT is the least

accurate method; it's simplistic and achieves a peak accuracy of just under 40%. SOGB shows an improvement of up to 10% over RT as is expected from the more complex modelling of the link CDFs. Both GA and SOGA achieve approximately 63% accuracy at their peak with only a small variation between them; this figure is greater than that for MOM (56%) although it occurs within a narrow range of δ. Given its greater complexity, SOGA may have been expected to perform better.

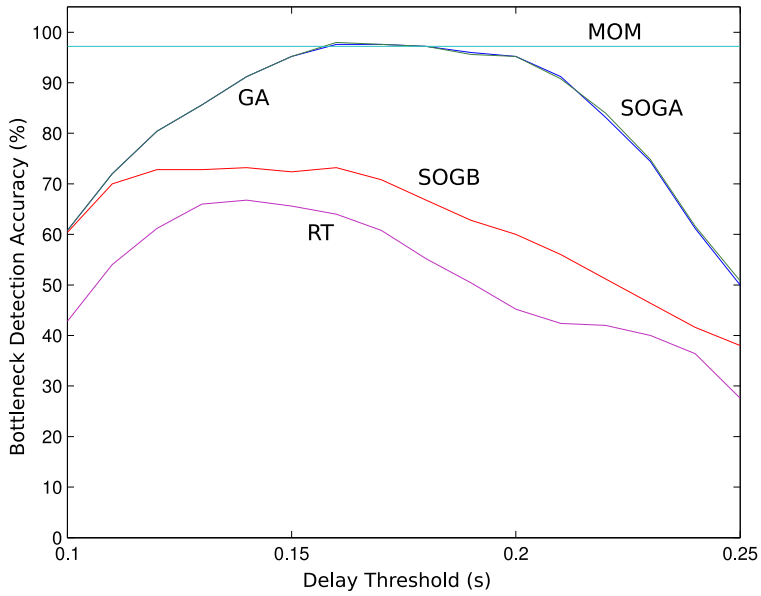


Figure 5.4: *Bottleneck link detection accuracy results for GA, MOM, RT, SOGA and SOGB for the 5 node topology with 50 ms separation.*

Figure 5.4 shows an increased separation, now 50ms, and an improvement in the accuracy of all methods. GA and SOGA have near identical accuracy and can outperform MOM's near constant accuracy of 97% although only by a slim margin at their peak; the peak range has increased to roughly twice that of Figure 5.3. RT remains the least accurate method with a peak accuracy of 65% whilst SOGB performs better with a peak of approximately 72%. The roll-off rate of GA and SOGA has decreased in comparison to Figure 5.3 while that of RT and SOGB has remained roughly constant; as before, MOM appears robust to the choice of δ.

Figure 5.5 shows the largest separation, 150ms, used in the 5 node topology. The best performing methods from the previous two figures, GA, SOGA and MOM exhibit 100% accuracy regardless of δ. This is not unexpected given the separation is large when compared to the delays of the non-bottleneck links. RT is the poorest performing method but achieves

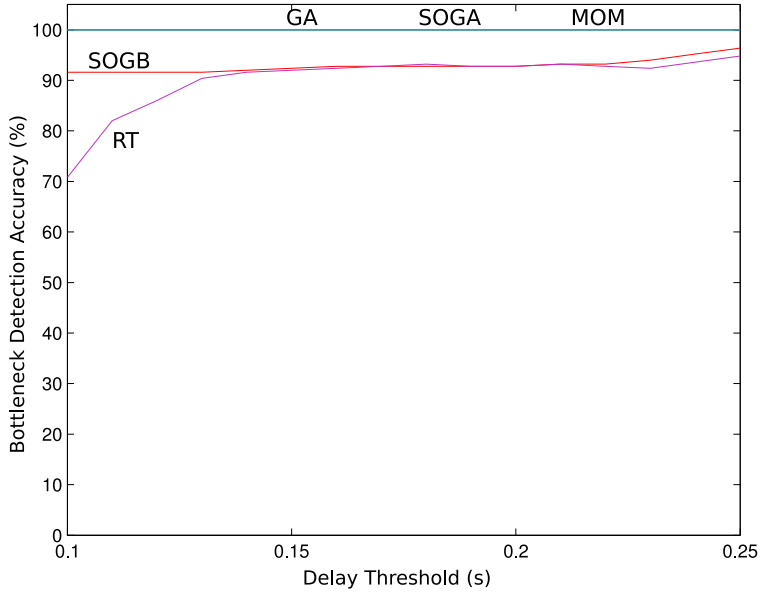


Figure 5.5: *Bottleneck link detection accuracy results for GA, MOM, RT, SOGA and SOGB for the 5 node topology with 150 ms separation.*

approximately 92% accuracy outside the lower tail, and approximately 95% at the upper tail. SOGB exhibits a similar performance to RT at the upper tail but does not drop below 92% at the lower tail.

These 5 node topology results exhibit the same trend as seen in the previous chapter. As the separation between the bottleneck link and the next worst link increases, so does the performance of the tomography methods. There appears to be little difference in the performance of GA and SOGA in these scenarios despite the increased complexity of SOGA.

Figure 5.6 shows results for the most challenging of the six scenarios used, the 10 node topology with 50ms separation. GA and MOM exhibit poor accuracy, consistent with observation from the previous chapter. RT and SOGB perform better with peak accuracies of 15% and 25% respectively. RT also exhibits performance over a greater range of δ than GA despite having a similar peak accuracy but is itself outperformed by SOGB in both peak accuracy, 25%, and range. SOGA exhibits the most surprising result, a peak accuracy of 95% which is in marked contrast to GA; both performed similarly in the 5 node topology. That said, there is a high sensitivity to δ with a steep roll-off.

The shape of the curves of the parametric methods in Figure 5.7 is similar to that in Figure 5.4

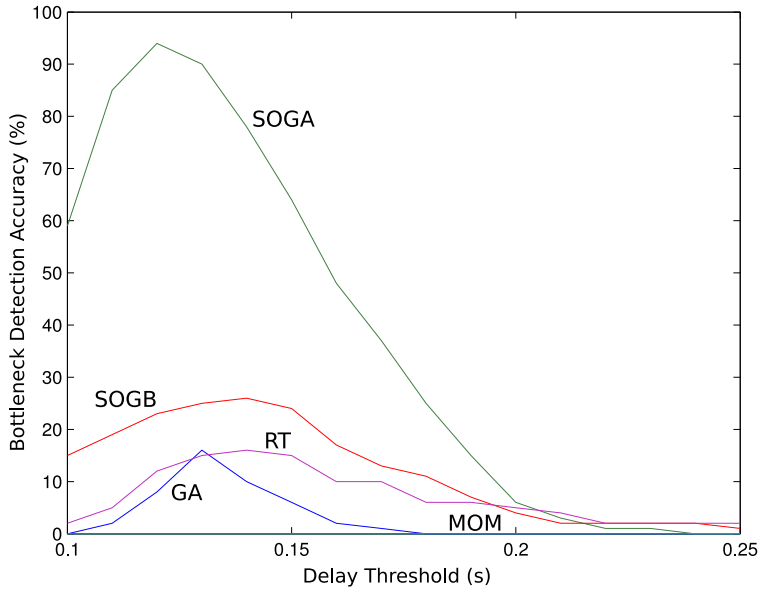


Figure 5.6: *Bottleneck link detection accuracy results for GA, MOM, RT, SOGA and SOGB for the 10 node topology with 50 ms separation.*

although here they have better accuracy at lower values of δ and both GA and SOGA achieve a peak accuracy of 100%. The order of SOGB and RT is as before with RT achieving a peak accuracy of 50% whilst SOGB achieves approximately 85%, both of which are more accurate than MOM. MOM is robust to changes in δ but achieves only 30% accuracy, an increase on the results shown in Figure 5.6.

Figure 5.8 shows all methods with an accuracy of 100% for part of the range of δ. SOGB experiences a slight reduction in accuracy at the lower values in the range with an accuracy of 97% at $\delta = 0.1$ which increases to 100% at $\delta = 0.13$. RT has an accuracy of 100% where $\delta \geq 0.2$ but below this is sensitive to the threshold and has a steep increase from an accuracy of 8% at $\delta = 0.1$ to its peak.

The 10 node topology results show a similar trend to that of the 5 node topology in that as the separation increases, so does the peak accuracy obtainable by each method. One notable difference is in the excellent performance of SOGA with a small separation; this was almost indistinguishable from that of GA in the smaller topology at all values of separation. Note that the previous top performer, GA has a lower accuracy than both RT and SOGB indicating a lack of robustness to the increased size of the network.

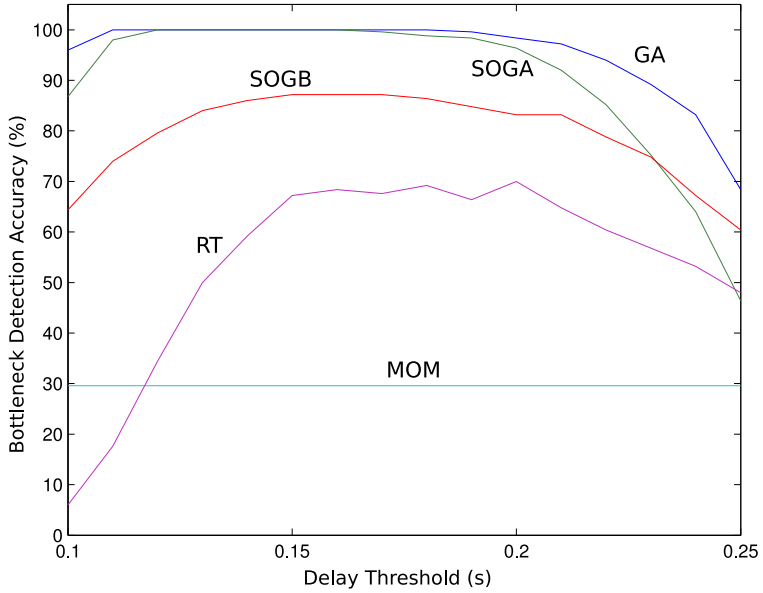


Figure 5.7: *Bottleneck link detection accuracy results for GA, MOM, RT, SOGA and SOGB for the 10 node topology with 100 ms separation.*

It may have been expected that the RT algorithm would achieve an accuracy higher than many of the other algorithms in these scenarios. It is simplistic in the sense that it does not modify the data or attempt to fit it to any distribution. Therefore, if it was assumed that the LS transform was ideal and introduced no errors then it would be expected to output a near-ideal link CDF representation - near-ideal as the binning process of the histogram step will introduce a small error. A near-ideal set of link CDFs coupled with a detection algorithm like CDFmax would be expected to achieve a high accuracy but instead the opposite is shown, a rather poor accuracy even with a large separation. As was discussed earlier (Section 2.4.1), the L2 pseudo-inverse used in LS is not perfect for this problem and here some evidence of that is seen. Based on the results above, it would not be unreasonable to assume the difference between the expected accuracy of RT and the measured accuracy can be attributed to errors introduced by LS. This error may be somewhat mitigated when using the parametric methods such as GA and SOGA as they employ an implicit averaging by extracting only statistics about the path data.

5.4.2 Complexity

Determining the complexity of the SOGA and SOGB algorithms in the same manner as in the previous chapter is difficult because of the use of successive approximation in fitting the

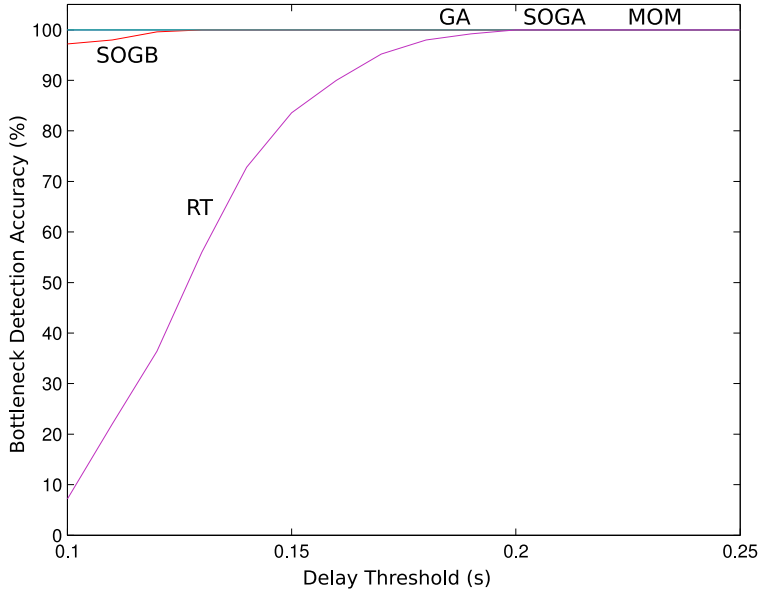


Figure 5.8: *Bottleneck link detection accuracy results for GA, MOM, RT, SOGA and SOGB for the 10 node topology with 250 ms separation.*

mixture models and therefore it is hard to give a good estimate of the number of operations required to perform an *average* estimation and detection. The metric used here is the mean time for the estimation and detection of 20 seconds worth of data. The mean is computed over three 20 second blocks with each block using a different value of δ to reduce the chances of any outliers caused by a dependency on the delay threshold. The results for both topologies are shown in Table 5.4. Specifications for the machine used to perform the work are to be found in Appendix A.

Method	Time - 5 node topology	Time - 10 node topology
GA	3.5 ms	10.0 ms
MOM	8.5 ms	58.0 ms
SOGA	268.0 ms	3740.0 ms
RT	3.1 ms	4.0 ms
SOGB	6.1 ms	11.0 ms

Table 5.4: *Mean computation times for the estimation and detection of one 20 second block of data using each of the five methods discussed in this chapter for both topologies.*

As could be expected, SOGA has a greater computational load than the other methods for both topologies - approximately two orders of magnitude greater than the next most complex method, MOM. This can be attributed to two factors: one, it employs a computationally

intensive successive approximation algorithm (EM) and two, it has a large number of combinations to compute in order to generate the output CDF. The effect of EM can be seen in SOGB which requires approximately twice the length of time that RT does; both algorithms perform the path estimation and transform stages in an identical manner so any differences can be attributed to the output distribution generation stage. In the previous chapter, an analytical result for GA suggested the difference in complexity between the 5 node and 10 node topologies was approximately one order of magnitude, whereas here it appears to be less. This can be attributed to a mismatch between the derivation of the analytical result and the executing of the code in MATLAB. The code will, almost certainly, undergo some form of optimization by MATLAB at runtime, parallelization and vectorization for example, which will result in observed mismatch. Another explanation, which will also contribute to the mismatch, is that the derivation is partly erroneous. The derivation was performed by hand for each method which may have also led to inconsistencies between methods.

5.4.3 Robustness

Robustness may be considered a useful metric when evaluating the different methods. A method may be efficient on a small scale, but to be useful it must remain efficient as the network size inceases. To gain an insight into the scaling properties, Table 5.5 shows the ratio of the computation time for the 10 node topology to the 5 node topology for all methods.

Method	**Ratio**
GA	2.86
MOM	6.82
SOGA	13.96
RT	1.29
SOGB	1.80

Table 5.5: *Ratios of the computation times of the 10 node to the 5 node topologies.*

It is expected that a robust method will have a ratio close to two, implying that it scales linearly with the number of nodes; some methods may have a ratio lower than two which would be an advantage in larger topologies. GA and SOGB have ratios close to two (2.86 and 1.80 respectively) while RT is lower at 1.29 indicating good robustness to scaling in these scenarios. SOGA is the least robust of all the methods with a ratio of to 14 which may arise from the non-linear increase in the size of the combination table from 3^5 to 3^{12} rows.

5.5 Conclusion

In this chapter three new methods for network tomography were introduced: RT, SOGB and SOGA. It was demonstrated that it is possible to extend existing methods to make use of the flexibility of Gaussian mixture models and that these can provide improved bottleneck detection accuracy although at a cost of computational complexity. Complexity and robustness to scaling were considered and it was shown that there is a non-linear increase in the complexity of GMM-based methods compared to the number of nodes in the network which poses a significant problem to using SOGB and SOGA in larger topologies.

Chapter 6
Extended methods

Introduction

This chapter is concerned with finding improvements to previously introduced methods. The improvements come in two areas: estimation algorithms and detection algorithms. In the area of estimation, two new algorithms are introduced, both based on the Pearson, a flexible family of four parameter distributions. One algorithm uses the Pearson type-1 distribution (PRS) in a parametric manner, similar to GA whereas the other algorithm combines it with MOM to create a hybrid-parametric algorithm. In the area of detection algorithms, the possibility of using the Kullback-Leibler divergence (KLD) as part of a detection algorithm to remove dependence on *a-priori* choice of the delay threshold, δ, is examined.

This chapter is divided into five sections. In the first section (6.1) a review of the key properties and definition of the Pearson distribution family is provided. The second section (6.2) introduces two new estimation algorithms based on the PRS. In the third section (6.3) a new approach to detection algorithms based upon the KLD is examined. In the fourth section (6.4) the parameters of the simulations used to provide results are defined before the results themselves are presented and discussed (6.5).

6.1 A review of some key properties of the Pearson distribution family

The PRS which is used later in this chapter comes from the Pearson family[1] of distributions which are a set of related distributions defined by four moments and therefore allow a greater flexibility in the shape of the distribution they describe than the two parameter distributions (i.e. the single Gaussian distribution) considered previously.

[1]This is may be referred to as the Pearson system in some literature but is identical. We use the Pearson family nomenclature in this thesis.

There are twelve defined distributions in the Pearson family and they were originally developed in order to model data with noticeable skew. Of the twelve distributions, one has already been covered in this work, the type V, of which the single Gaussian distribution is a special case. It is named in the Pearson system to show its relation to the other types. Other known distributions such as the Beta prime distribution (type VI) and the gamma distribution (type III) come from the Pearson system and later acquired their now more common names.

All of the distributions in the Pearson family can be described using two values, β_1 and β_2. β_1 is defined as the square of the skewness whereas β_2 is defined as the *traditional* kurtosis. Kurtosis is more normally described either using cumulants or as the fourth standard moment about the mean divided by the variance squared, minus three. The correction by 3 is to ensure that the kurtosis of a single Gaussian distribution is equal to 0. Clearly this leads to ambiguity when describing β_2 so to avoid any such ambiguity or confusion, (6.1) describes the skewness and (6.2) describes the kurtosis as computed in this work. This allows β_2 to be defined as $\beta_2 = \gamma_2 + 3$.

$$\gamma_1 = \frac{\kappa_3}{\kappa_2^{3/2}} = \frac{\mu_3}{\sigma^3} \tag{6.1}$$

$$\gamma_2 = \frac{\kappa_4}{\kappa_2^2} = \frac{\mu_4}{\sigma^4} - 3 \tag{6.2}$$

$$\widehat{\mu_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} Y_{ik} \tag{6.3}$$

$$\widehat{\sigma^2_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} (Y_{ik} - \widehat{\mu_{Y_i}})^2 \tag{6.4}$$

$$\widehat{\gamma_1} = \frac{1}{N}\sum_{k=1}^{N} \frac{E((Y_{ik} - \mu_{i,1})^3)}{\sigma^3} \tag{6.5}$$

$$\widehat{\gamma_2} = \frac{1}{N}\sum_{k=1}^{N} \frac{E((Y_{ik} - \mu_{i,1})^4)}{\sigma^4} - 3 \tag{6.6}$$

6.2 Estimation and detection procedures

In this section two new methods for network tomography which make use of the PRS are introduced. The first is based solely upon the PRS whilst the second is a modification of MOM to use the PRS as an output stage forming a hybrid-parametric algorithm.

6.2.1 The Pearson type-1 distribution

In the GA algorithm, only the first two moments, the mean and variance, are used to model the delay distributions on both links and paths. It is suspected that this approach may not offer enough flexibility to provide an accurate model so, to increase flexibility, it is possible to use a Pearson type-1 distribution which uses the first four moments.

To begin with, each of the four moments are estimated for the data measured on each path using the relevant equations: 6.3, 6.4, 6.5 and 6.6 where i denotes the path in question. In a similar fashion to GA, each moment is transformed using LS to provide an estimate of the moment for each link. The estimated link mean and variance are as shown in (6.7) and (6.8) whilst the estimated skewness and kurtosis are shown in (6.9) and (6.10) respectively. These estimates can be used to define a set of Pearson type-1 distributions which represent the estimated packet delay distributions on each of the links. This algorithm, as shown in Algorithm 6, is similar to that of GA but with the inclusion of skewness and kurtosis.

$$\widehat{\mu_{X_j}} = h_{ij} \cdot \sum_{i=1}^{P} \mu_i \tag{6.7}$$

$$\widehat{\sigma^2_{X_j}} = |h_{ij}|^2) \cdot \sum_{i=1}^{P} \sigma_i^2 \tag{6.8}$$

$$\widehat{\mu_{3_{X_j}}} = h_{ij} \cdot \sum_{i=1}^{P} \mu_{3_i} \tag{6.9}$$

$$\widehat{\mu_{4_{X_j}}} = h_{ij} \cdot \sum_{i=1}^{P} \mu_{4_i} \tag{6.10}$$

Algorithm 6 PRS estimation algorithm

1: **for** $i = 1$ to P **do**
2: fit a Pearson type-1 distribution to path i using equations (4.2), (4.3), (6.1) and (6.2).
3: **end for**
4: **for** $j = 1$ to L **do**
5: generate a PRS CDF for link j using estimated quantities as computed from path estimates as given by equations (6.7), (6.8), (6.9) and (6.10).
6: **end for**

Practically, a Pearson type-1 distribution is often estimated using a four parameter Beta distribution and the implementation used in this thesis was subject to the condition: $\gamma_2 > \gamma_1^2 + 1$. There were a small number of cases in which this condition was not satisfied. To remedy this, γ_2 was set to be $\gamma_1^2 + 1 + 0.0001$. The additional 0.0001 was inserted to ensure the condition was not further broken due to any rounding.

6.2.2 The Pearson method-of-moments

One problem encountered when using MOM is that it does not produce an estimated CDF of the delay distribution on each link but instead a CGF. Having the estimated link distribution represented as a CDF is preferable to having it represented as a CGF as it may give more insight into the operation of the network and it is easier to manipulate. Conversion between a CGF and a CDF is non-trivial as the CGF may have an unbounded number of cumulants and therefore mapping to a distribution with a fixed number of non-zero cumulants is not possible. To overcome this problem, it is possible to combine two previously seen methods, MOM and PRS to generate the Pearson method-of-moments.

Initially, MOM is used to estimate the first four moments of the delay distribution on each link; this is the standard MOM algorithm with $t = 4$. The cumulants are then converted to moments using standard manipulations; the four estimated moments can then be used as the basis for defining a Pearson type-1 distribution CDF for each link. The equations (6.11), (6.12), (6.1) and (6.2) give formulae the conversion between the first four cumulants and corresponding moments. This algorithm has the advantage of combining the robust estimation methods of MOM, which have been observed to offer a desirable insensitivity to δ, with a desirable CDF output. Algorithm 7 shows the estimation algorithm in the normal format.

$$\widehat{\mu_{X_j}} = \kappa_1 \tag{6.11}$$

$$\widehat{\sigma^2_{X_j}} = \kappa_2 \tag{6.12}$$

Algorithm 7 P-MOM estimation algorithm

1: **for** $i = 1$ to L **do**
2: estimate CGF of path i using equation 4.6
3: **end for**
4: **for** $j = 1$ to L **do**
5: estimate CGF of link j using path estimates and LS as in equation (4.7)
6: **end for**
7: **for** $j = 1$ to L **do**
8: convert estimated cumulants for link j into estimated moments using equations (6.11), (6.12), (6.1) and (6.2).
9: generate a Pearson type-1 distribution CDF for each link, j, using the estimated moments
10: **end for**

6.2.3 Detection methods

Both the PRS algorithm and the Pearson method-of-moments (P-MOM) algorithm produce a CDF output so it is possible to couple the CDFmax detection algorithm with them to form the PRS method and the P-MOM method. The CDFmax algorithm was introduced in section 4.2 and is used unchanged.

6.3 A new detection algorithm

Two detection algorithms are used in conjunction with the estimation algorithms in this thesis: CDFmax and the Chernoff bound. They both perform well for their respective class of estimators but are reliant on the *a-priori* selection of the delay threshold parameter, δ. This leads to a problem when considering the implementation, of those methods which use them, in hardware: a value must be chosen for δ to allow comparisons to take place and bottlenecks to be detected, but, the value can only be chosen empirically by observing the network under test for a period of time and accumulating some knowledge about the range of the delay, i.e. when it is *too high* and causing a bottleneck. A preferable approach would be to use a detection algorithm which requires no parameters to be chosen *a-priori* and which can operate with the CDFs output from the estimation algorithms. One class of algorithm which is well suited to the problem of defining the separation of distribution functions is the f-divergence.

The f-divergences (FDs) [84] are a class of algorithm commonly used in information theoretical work; their basic form and theory was introduced in Section 2.4. Here they are examined in more detail.

6.3.1 The Kullback-Leibler divergence

The most widely known and used of the FDs is the KLD [12] and its most common use is to provide a numerical answer to the question: *How different are two PDFs?* The KLD is defined as either the divergence between either two PDFs (6.13) or two probability mass functions (PMFs) (6.14). For continuous distributions such as the Gaussian distribution, a compact closed form solution exists therefore computing the integral is unnecessary. This is not true of all distributions and to provide compatibility with the existing algorithms and in order to handle any estimated output CDF in a computationally efficient manner, the discrete form is used in the remainder of this work. This implies that any estimated CDFs are converted to PMFs; this is a simple manipulation and involves sampling the CDF and normalising to ensure a valid PMF.

$$D_{\mathrm{KL}}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx \tag{6.13}$$

$$D_{\mathrm{KL}}(P||Q) = \sum_{i} P(i) \log \frac{P(i)}{Q(i)} \tag{6.14}$$

6.3.2 Kullback-Leibler divergence: A synthetic example

The easiest method to demonstrate the use of the KLD as a detection algorithm is with a small example. First, consider a case where four link PDFs have been estimated using the GA algorithm and have parameters as shown in Table 6.1. Link 4 is the bottleneck link with a mean delay of 0.4 s and a separation of approximately 0.28 s. The CDFmax algorithm would find link 4 to be the bottleneck with a very high accuracy in this scenario given the high separation relative to the mean delay of the links. The link CDFs and PMFs are shown in Figures 6.1 and 6.2 respectively.

Using the PMFs and the KLD algorithm it is possible to compute a table showing pairs of links and the divergence between them; Table 6.2 is one such table and shows the pairs and divergences for the link used in this example. What is noticeable from the table is that each pair

Link number	μ	σ^2
1	0.10	6.25×10^{-4}
2	0.08	1×10^{-4}
3	0.12	1×10^{-4}
4	0.40	4×10^{-4}

Table 6.1: *Statistical properties of the four links used in the KLD example.*

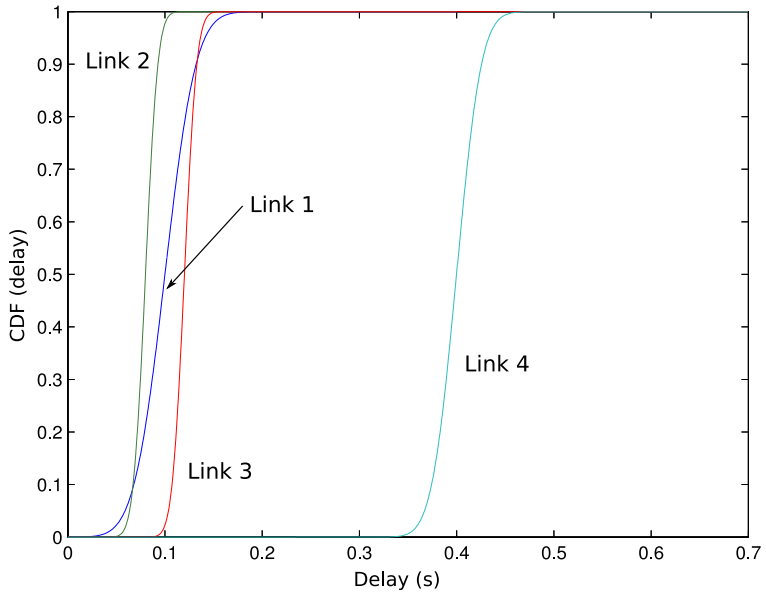


Figure 6.1: *Four synthetic link CDFs used in the KLD example. Link 1 =* $\mathcal{N}(0.1, 6 \times 10^{-4})$*, Link 2 =* $\mathcal{N}(0.08, 1 \times 10^{-4})$*, Link 3 =* $\mathcal{N}(0.12, 1 \times 10^{-4})$*, Link 4 =* $\mathcal{N}(0.4, 4 \times 10^{-4})$*.*

which includes link 4 has a large divergence relative to the other pairs. If it is assumed that the link PDFs are reasonably similar in shape it would also be reasonable to choose link 4 as the probable bottleneck in this scenario because of its separation from the other links. However, were it not reasonable to assume the shape of the PDFs was similar then this would cause a problem.

6.3.3 Kullback-Leibler divergence: A real example

Consider next Figures 6.3 and 6.4 where the output of the GA estimation algorithm is plotted in both CDF and PMF form respectively. The data comes from the 5 node network scenario with a 20ms separation and the parameters are shown in Table 6.3. From the CDF it can be seen that

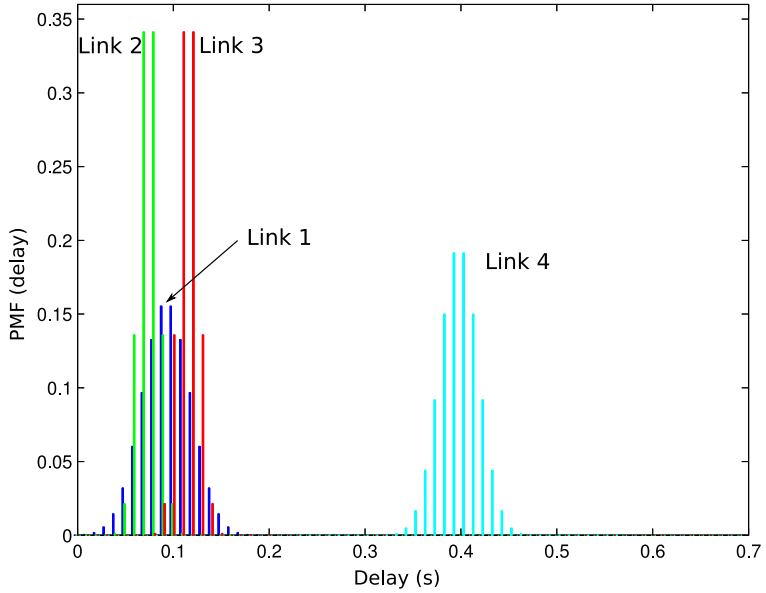


Figure 6.2: *Four synthetic link PMFs used in the KLD example. Link 1 =* $\mathcal{N}(0.1, 6 \times 10^{-4})$*, Link 2 =* $\mathcal{N}(0.08, 1 \times 10^{-4})$*, Link 3 =* $\mathcal{N}(0.12, 1 \times 10^{-4})$*, Link 4 =* $\mathcal{N}(0.4, 4 \times 10^{-4})$*.*

Link pair	KLD	Link pair	KLD
1,2	3.0159	2,3	10.7331
1,3	3.0161	2,4	49.4166
1,4	49.4166	3,4	48.7791

Table 6.2: *Link pairs and their respective KLDs for links used in the KLD example.*

link 1 is the probable bottleneck but this is much harder to discern using the PMF alone. If the KLD is applied as above to the links in pairs then the results shown in Table 6.4 are observed.

What can be seen from this data is that where the separation is low using only the KLD as a detection algorithm can lead to incorrect results. In this example, link 3 appears to be the *most different* from the others, however, it is link 1 that is the bottleneck.

6.3.4 Kullback-Leibler divergence: Conclusion

Whilst it overcomes one of the main problems of CDFmax, the necessity for *a-priori* selection of the delay threshold, the KLD appears not to be robust enough to be a direct replacement. The main problem with KLD is that it detects the total divergence between the two PMFs it

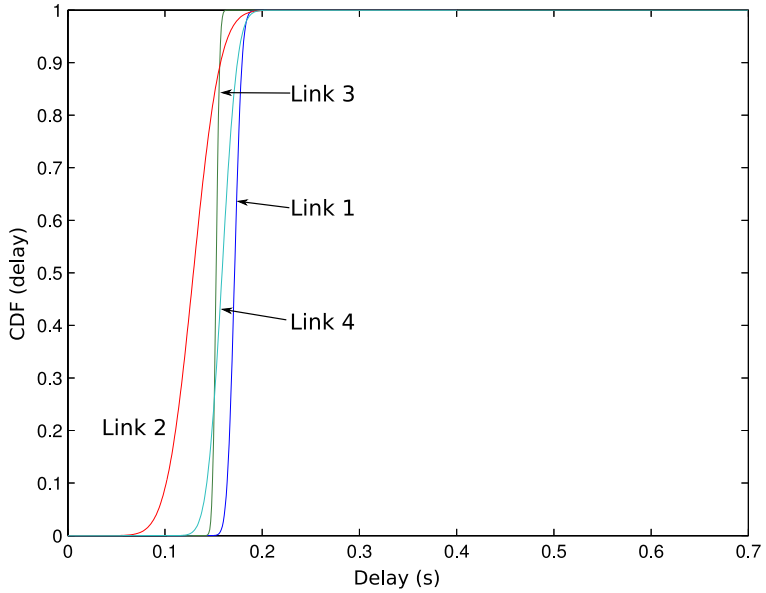


Figure 6.3: *Four estimated link CDFs. Link 1 =* $\mathcal{N}(0.1716, 0.0060)$*, Link 2 =* $\mathcal{N}(0.1526, 0.0029)$*, Link 3 =* $\mathcal{N}(0.1292, 0.0218)$*, Link 4 =* $\mathcal{N}(0.1587, 0.0123)$*.*

Link number	μ	σ^2
1	0.1716	0.0060
2	0.1526	0.0029
3	0.1292	0.0218
4	0.1587	0.0123

Table 6.3: *Statistical properties of four links estimated using the GA algorithm from a 5 node topology with 20ms separation.*

operates on and therefore if they are of rather different shape, the divergence will be high. This is overcome in cases where the separation is large as the divergence due to the separation outweighs the divergence due to shape mismatch. Perhaps the most useful application is therefore to initially spot anomalous links which require processing with CDFmax for more detail.

6.4 Simulation details

As with previous chapters, the six scenarios that were introduced in Chapter 4 are used as a basis to compare the accuracy of the methods. It was previously seen that GA is a robust

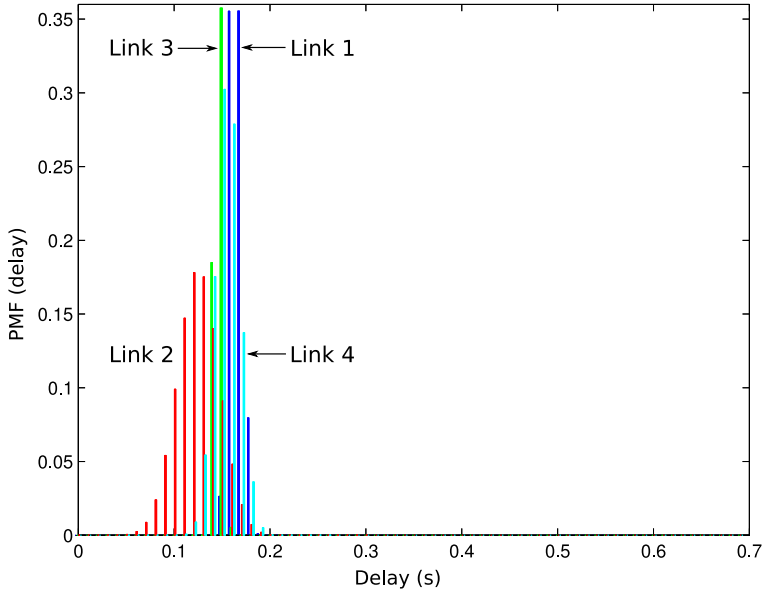


Figure 6.4: *Four estimated link PMFs. Link 1 =* $\mathcal{N}(0.1716, 0.0060)$*, Link 2 =* $\mathcal{N}(0.1526, 0.0029)$*, Link 3 =* $\mathcal{N}(0.1292, 0.0218)$*, Link 4 =* $\mathcal{N}(0.1587, 0.0123)$*.*

Link pair	KLD	Link pair	KLD
1,2	13.5303	2,3	15.1732
1,3	15.7686	2,4	4.5287
1,4	2.4873	3,4	3.1140

Table 6.4: *Link pairs and their respective KLDs for four links from a 5 node network scenario with 20ms separation. The link PDFs are estimated using the GA estimation algorithm.*

parametric method and MOM is a robust non-parametric method and therefore they are used as benchmarks by which to judge the performance of PRS and P-MOM. The inclusion of MOM makes it possible to see if the accuracy of P-MOM follows that of a parametric or non-parametric method. The parameters for MOM and GA are those given in Section 4.3.

6.5 Simulation results

As in the previous two chapters, the performance of the four methods presented in this chapter is examined by considering bottleneck detection accuracy, computational complexity and robustness to scaling. As noted in previous chapters, the *separation* is defined as the difference

in delay between the bottleneck link and the next worst link.

6.5.1 Bottleneck link detection accuracy

Figure 6.5 shows the 5 node topology with a separation of 20ms. GA and MOM have the same response as in previous chapters which gives a benchmark for comparing the other methods. It is clear that PRS outperforms the other three methods with a peak accuracy approximately 20% higher than that of GA. PRS also has a higher accuracy than GA across the range of δ and a higher accuracy than MOM and P-MOM when $\delta \leq 0.18$. The response of P-MOM mimics that of MOM but with an increased variance which is attributed to the reduced number of moments used to estimate the distributions in P-MOM, when compared with MOM. Over the range of δ plotted, a slight downward trend is observed as δ increases bringing the accuracy down from 60% at $\delta = 0.1$ to 50% at $\delta = 0.25$. This could be an exaggeration of the slight variation observed in MOM with respect to δ.

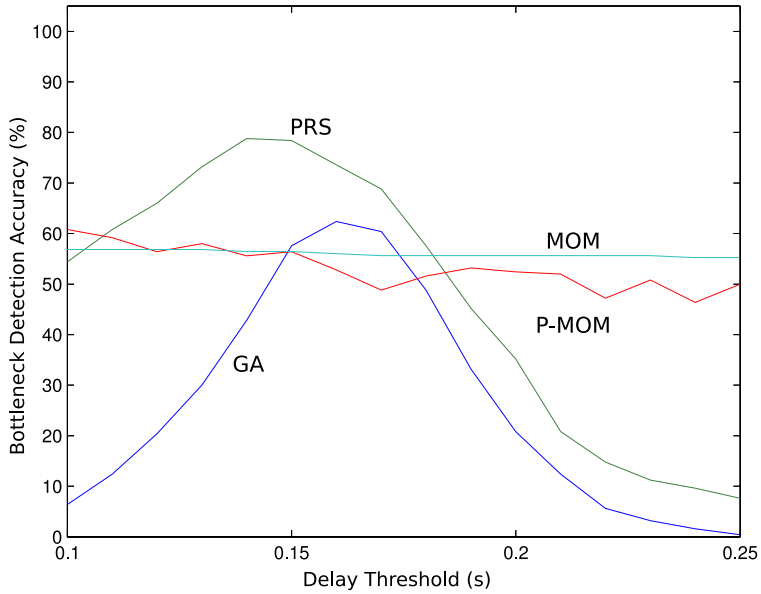


Figure 6.5: *Bottleneck link detection accuracy results for GA, MOM, PRS and P-MOM for the 5 node topology with 20 ms separation.*

Figure 6.6 shows the 5 node topology with an increased separation of 50ms. In contrast to the the previous scenario, the ordering of the response curves is reversed. PRS now has the lowest peak accuracy at 92% (which is still an increase compared to the results seen in Figure 6.5) and is the poorest performer over most of the range of δ plotted. The steep roll-off of GA

at the lower tail gives it the poorest performance when $\delta \leq 0.12$. The peak accuracy of GA is equal to that of MOM; P-MOM responds in a similar manner to the previous scenario: a similar accuracy to MOM but with an increased variance and a downward trend as δ increases. However, in this scenario, the trend is reduced: a fall of approximately 5% from 98% to 93% over the range of δ plotted.

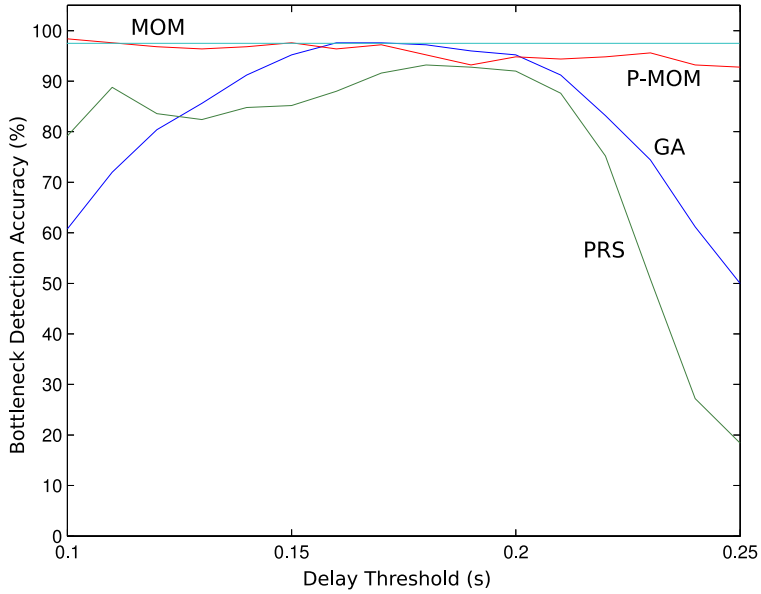


Figure 6.6: *Bottleneck link detection accuracy results for GA, MOM, PRS and P-MOM for the 5 node topology with 50 ms separation.*

In the final 5 node scenario, the separation is increased to 150ms where the response of all four methods is 100% across the range of δ. P-MOM does not exhibit its normal variation, which can be attributed to the large separation.

Figure 6.7 shows the larger, 10 node topology, with 50ms separation. As was seen in previous chapters, MOM has a response of 0% accuracy, however, P-MOM exhibits no variance, which provides some evidence to support the hypothesis that at the limits, the variance of P-MOM tends to zero. The GA and PRS responses have an order similar to that seen in Figure 6.5. PRS has a higher peak accuracy of just over 20% compared to 15% for GA and also a higher accuracy over the range of δ plotted.

Figure 6.8 shows the 10 node scenario with an increased separation of 100ms. GA and PRS exhibit a similar inversion of performance as seen in Figure 6.6 with GA having a peak accuracy of 100%, outperforming the peak accuracy of PRS by 3%. The shape of the PRS repsonse is

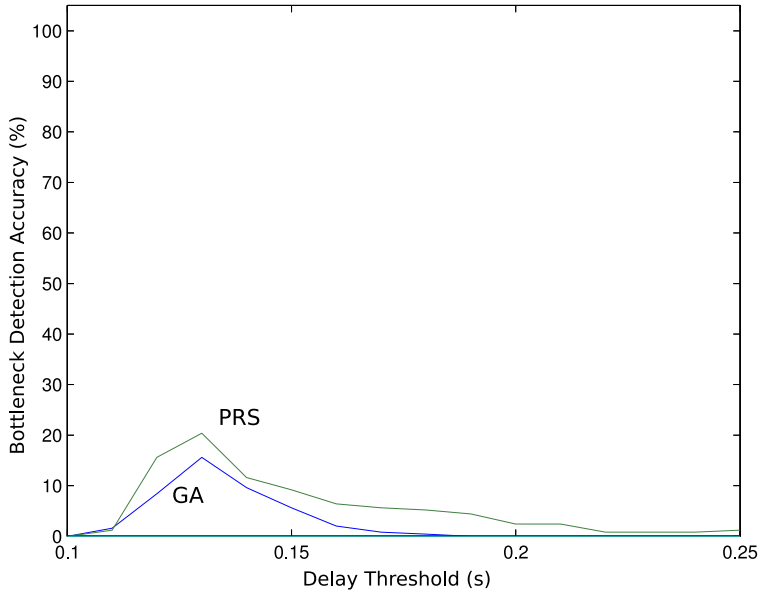


Figure 6.7: *Bottleneck link detection accuracy results for GA, MOM, PRS and P-MOM for the 10 node topology with 50 ms separation.*

markedly narrower than that of Figure 6.5. The response of MOM and P-MOM exhibit the same differences as seen previously, MOM with an accuracy of 29% across the range of δ plotted and P-MOM with a similar general response but with a variance of up to 5%. Unlike in previous work, no downward trend in the accuracy of P-MOM as δ increases is seen, instead there seems to be a decrease in the variance.

Figure 6.9 shows the 10 node topology with a large separation of 250ms. As expected from previous work, the response of GA and MOM is 100% accuracy across the range of δ plotted. The response of P-MOM is unusual in that it has a lower mean than than of MOM, 98% compared to 100%, but has no noticable variance. It could be expected to either have a mean of 100% confirming the hypothesis that at the limits of accuracy the variance of P-MOM tends to zero, or to have a mean which decreases as δ increases but with some observable variance. It would not now be unreasonable to hypothesise that the variance of P-MOM is related to separation and that the sensitivity decreases as the separation increases which results in a lower variance. The response of PRS is unusual as it exhibits a marked sensitivity and decrease in accuracy where $\delta \leq 0.13$ which is very similar to its response in Figure 6.7 where the separation is much lower.

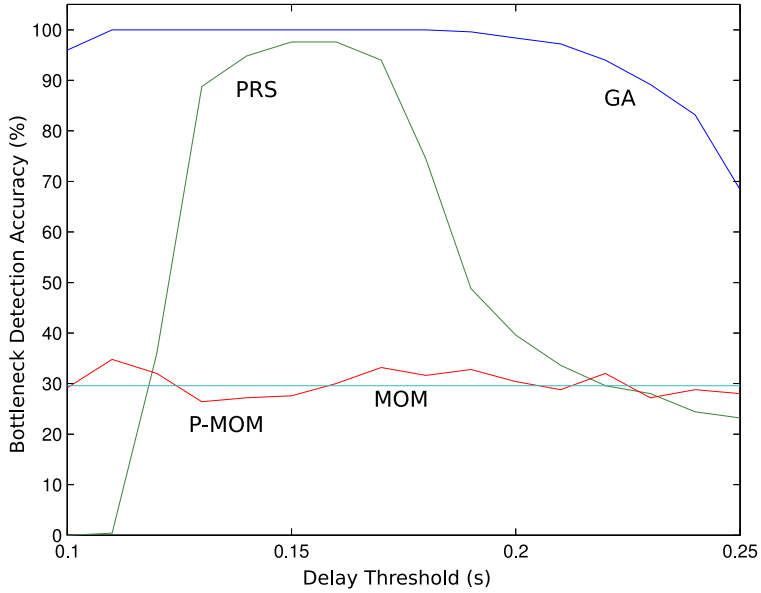


Figure 6.8: *Bottleneck link detection accuracy results for GA, MOM, PRS and P-MOM for the 10 node topology with 100 ms separation.*

6.5.2 Computational Complexity

The four methods are compared by considering their algorithmic representations (Table 6.5) and the number of operations required to process one 20s block of data (Table 6.6). The formulae are derived from the implementations used in the thesis and improvements could be made via more efficient implementations.

The parametric methods, GA and PRS, have a number of ADDs which scales with PN. PRS is burdened by of a constant of $11P$ compared to the much smaller $2P$ of GA. A similar result is seen with the number of MULTs. The $3PN$ of GA contrasts with the $8PN$ of PRS. Whilst this may appear a small increase, N can scale quickly as the length of the observed data increases. Even a small multiplier attached to N can significantly increase the computational load.

Comparing MOM and P-MOM, the most noticeable difference is that MOM is dependant on the CGF parameter t with regards to the number of ADDs and MULTs. P-MOM is dependant on t, but as t is fixed at 4, this term appears as a constant and not a parameter. Both MOM and P-MOM scale by a constant of L in the number of MULTs and by $(L-1)!$ in the number of ADDs. This is unsurprising given the similarities in their estimation processes. What can be seen in all methods is that N, the number of samples, dominates parameter values in all but

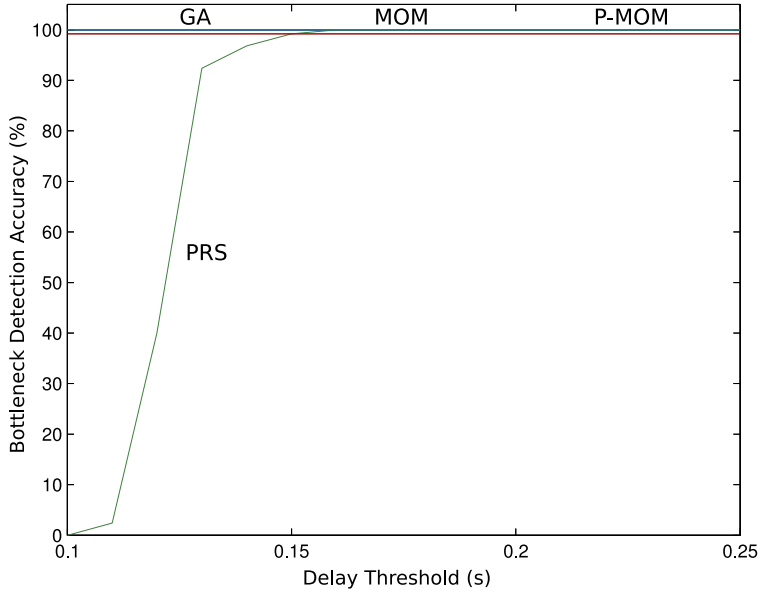


Figure 6.9: *Bottleneck link detection accuracy results for GA, MOM, PRS and P-MOM for the 10 node topology with 250 ms separation.*

small block sizes, where N would be small. For the block size of 20s used in these examples, $N = 128$. For a block size of 100s, N rises to 640.

Method	**MULT**	**ADD**
GA	$L(87 + 3P) + P(N + 2)$	$30L + L(P - 1) + 3PN$
PRS	$P(11N + 5L + 2) + 165L$	$8PN + 4L(P - 1) + 52L + (L - 1)!$
MOM	$Pt(2N + 1) + L(5t + 2)$	$PtN + LP + (L - 1)!$
P-MOM	$167L + 4P(N - 1)$	$L(P + 52) + 4PN + (L - 1)!$

Table 6.5: *Formulae for number of MULT and ADD operations required for the estimation and detection of N samples of data in a $L + 1$ node network comprising L links and P paths.*

In addition to examining the algorithmic forms and equations for these methods, some insight can also be gained by looking at numeric results. Table 6.6 shows the number of operations required for each of the four methods using the parameters as given in Table 4.1.

What is seen is that the number of operations used by MOM much greater than that used by P-MOM. This is despite the relationship between the CGF parameter, t, being five - $t = 20$ for MOM and $t = 4$ for P-MOM. P-MOM combines the estimation stage of MOM with the output stage of PRS. It uses approximately half the number of ADDs and MULTs that PRS does and approximately a tenth of the number of MULTS and a fifth the number of ADDs that MOM

does.

Considering the set of parametric methods, PRS is more complex than GA by a factor which outweighs the number of parameters estimated. The number of both ADDs and MULTs used by PRS is between two and a half and eight times that of GA despite PRS estimating twice the number of parameters that GA does. The scaling in both cases is non-linear with respect to the number of parameters which must be estimated.

Method	**MULT**	**ADD**
GA	1058	2056
PRS	7810	5398
MOM	26,108	12,826
P-MOM	3208	2794

Table 6.6: *Numerical results for the formulae in Table 6.5 for the 5 node topology with 20ms separation with* $N = 1000$ *and other values as in Table 4.1.*

To compare these methods with the GMM-based methods, consider Table 6.7 which shows the computation time required by each of the methods examined in this thesis. The scenarios used are the 5 node topology with 20ms separation and the 10 node topology with 50ms separation. The time taken is to compute the estimation and detection algorithms for a 20s block of data.

What is noticeable is that both PRS and P-MOM have a much higher computation time than would be expected by examining their algorithms. In the 5 node topology, the computation time for PRS is higher than that of SOGA which was previously assumed to be the least efficient method. In the 10 node topology, SOGA is the least efficient but the times for PRS and P-MOM are still higher than would be expected

One explanation for this discrepancy is that the implementation of the algorithms using the Pearson type-1 distribution is not particularly efficient, much less so than the GMM-based methods. Should a more efficient implementation be used then it is expected that the computation times would fall bringing them closer to the order of magnitude of SOGB, MOM and GA.

6.5.3 Robustness to scaling

Related to the computational complexity of the methods and their detection accuracy is the behaviour when scaling. A method may offer a desired accuracy at an acceptable level of

Method	Time - 5 node topology	Time - 10 node topology
GA	3.5 ms	10.0 ms
MOM	8.5 ms	58.0 ms
SOGA	268.0 ms	3740.0 ms
RT	3.1 ms	4.0 ms
SOGB	6.1 ms	11.0 ms
PRS	289.9 ms	1617.0 ms
P-MOM	139.1 ms	583.9 ms

Table 6.7: *Mean computation times for the estimation and detection of one 20 second block of data using all methods discussed in this thesis for both topologies.*

computational complexity for one size of network however, if the complexity scales faster than the network size does or if the accuracy changes as the network changes size then it may not be regarded as useful. It is possible to catch a glimpse of the effects of scaling by comparing the ratio of the number of MULT and ADD operations for each method for both topologies studied so far.

Method	MULT	ADD
GA	2.42	2.40
PRS	2.40	3.41
MOM	2.40	2.80
P-MOM	2.40	4.39

Table 6.8: *The ratios of the number of operations required to compute one block of data in the 10 node topology to the number of operations required to compute one block of data in the 5 node topology for GA, MOM, PRS and P-MOM.*

From the results in Table 6.8 two things can be seen: one, the number of MULTs scales at a rate equal to that of the paths ($12/5$) whereas the number of ADDs scales at a higher rate and two, the lowest of the ratios is greater than the ratio of the number of nodes, $N_{\text{nodes}} = L + 1$. Given the ratios of P and L, a method which scales in a robust fashion should have a number of operations (both ADD and MULT) with a similar ratio. Of the four methods considered, GA and MOM are closest to this number, the two other methods, PRS and P-MOM are furthest from it - interestingly, these both use the Pearson distribution. In a network where $1 \leq P/L \leq 1.5$ and where the number of nodes is 100 or 1000 then a small deviation from the ideal upper bound (the largest of the ratios of P or L) at this stage results in many more operations being required to process the data. In a real-world real-time scenario this may limit the scope of operation of the method.

6.6 A summary of methods

In this and the previous two chapters a number of different methods for network tomography have been introduced, analysed and the results compared using different metrics. It is therefore appropriate at this point to provide a summary of the key features of each method and to consider performance across the set.

The simplest methods in terms of the algorithmic complexity, GA and RT are also those with lowest computational complexity and lowest computation time when implemented. The performance of RT was generally the least accurate of all methods in each scenario except the 10 node topology with 50ms separation in which it outperformed GA and MOM (see Figure 5.4). RTs performance was limited by the L2 pseudo-inverse transform and *a priori* it could have been expected to be the most accurate of all methods due to the low volume of processing and modification it performed on the measured data. GA performed better often achieving a level of accuracy higher than more complex methods but was seen to have a dependence on the value of the delay threshold δ. It had both a low algorithmic and computational complexity combined with a low computation time which makes it competitive with RT in a scenario where computational power and time are low but a level of accuracy which is less than 100% is acceptable.

Two methods, SOGA and SOGB, which used a GMM were also considered. SOGB was perhaps the most simplistic of these as it used the same mechanism as RT but instead of a raw histogram output, it generates a CDF using a GMM fitted to the raw histogram. This keeps computational time low - it is of the same order as RT and GA. However, due to the GMM fitting process no algorithmic complexity could be calculated. Accuracy was in general better than that of RT, and it appears reasonable to assume that this was the result of the smoothed CDF output although it too displayed sensitivity to δ as with RT and GA. The other GMM-based method, SOGA, was an extension of the GA method. It assumed that each link delay distribution could be modelled by a single Gaussian distribution and therefore the path delay distribution could be modelled by a GMM. By using what was essentially a clustering algorithm, an estimate for each link could be derived. The accuracy was in many scenarios slightly worse than that of GA perhaps as a result of over-fitting and also sensitive to δ. This was disappointing given the high computation time which could be attributed to computing the large number of combinations for each link and fitting with the iterative EM algorithm. In one scenario, the 10 node topology with 50ms separation, which was seen previously to be a challenging scenario, this method

performed very well with an accuracy much in excess of the other methods.

The other three methods, MOM, P-MOM and PRS can be considered together. MOM was one of the suggested algorithms in [1]. It has a moderate computational complexity and the computation time was slightly under three times that of RT for the 5 node topology. It exhibited two features which set it apart from the parametric methods: one, it showed very little sensitivity to δ and two, it appeared to have a lower separation threshold below which the accuracy dropped off markedly. With sufficient separation however, its performance was excellent and the lack of variation with δ makes it a useful candidate where high accuracy is required but δ cannot be estimated *a priori*. PRS was an adaptation of the GA method but replacing the single Gaussian distribution with a Pearson type-1 distribution. This increased the computational complexity and computation time greatly, the latter possibly due to poor implementation and not directly as a result of increased complexity of the algorithm. Accuracy was better than that of GA in scenarios with low separation such as the 5 node topology with 20ms separation and the 10 node topology with 50ms separation. As the separation increased, PRS exhibited a slower increase in accuracy than GA, with poorer accuracy in scenarios such as the 5 node topology with 50ms separation. The final method, P-MOM was an attempt to combine the PRS output stage with the MOM estimation stage to reduce complexity and provide a way to use a simple detection algorithm such as CDFmax with MOM. The results were interesting, P-MOM had a response similar to that of MOM in that it exhibited a lower separation threshold and a low sensitivity to δ but did exhibit some variance (see Figure 6.5). The computational complexity appears low but the computation time is far higher than would have been expected, probably due to inefficient implementation.

6.6.1 Bottleneck Link Placement

In this, and the previous two chapters, the bottleneck link was fixed to be one specific link. This allowed a comparison between tomography methods for a range of previously defined scenarios, with all other network parameters held constant. In a real network however, the bottleneck link is unlikely to be fixed as one specific link. To observe the bottleneck link detection accuracy of the methods when the bottleneck link is moved, a further simulation was performed. This used the 5 node topology with 50ms separation, and the bottleneck link was fixed as link 2. All other network parameters are as defined previously - see sections 4.3 and 5.3 for parameter values. The bottleneck link detection accuracy results for MOM, GA and RT

are plotted in Figure 6.10.

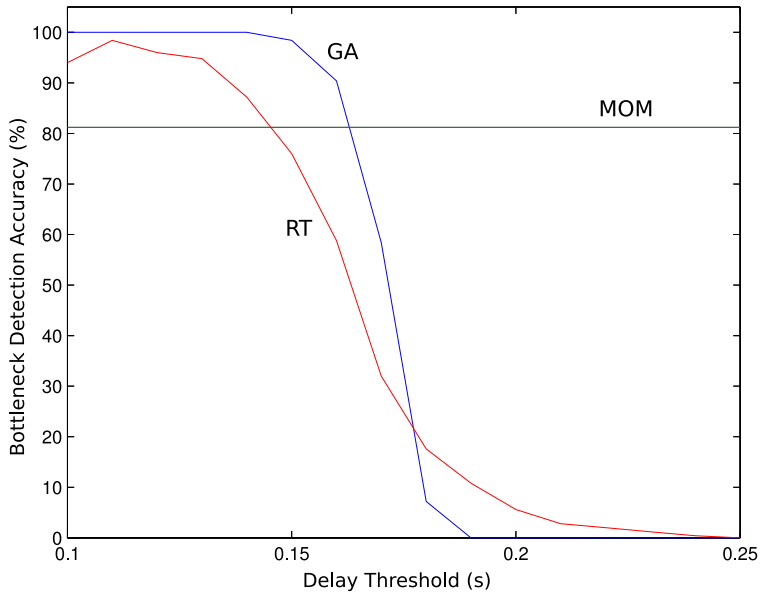


Figure 6.10: *Bottleneck link detection accuracy results for GA, MOM and RT for the 5 node topology with 50 ms separation where link 2 is the bottleneck link.*

What is seen, is that the location of the bottleneck node has an effect on the detection accuracy of the tomography methods. The measurement, estimation and detection methodology remains valid. Each method is able to detect the bottleneck link, in some cases with a high degree of confidence, for some values of the delay threshold. The methods show trends in detection accuracy which are comparable with those seen in previous scenarios, where link 1 was the bottleneck link. MOM has a detection accuracy which varies very little as the delay threshold is varied. GA and RT are sensitive to the delay threshold. In this case, the sensitivity is quite marked, with accuracy falling dramatically where $\delta > 0.15$. The relative performance of both is consistent with that seen earlier, for example, in Figure 5.4, where RT has lower accuracy than GA. However, there is a reversal of the performance at higher values of δ, RT has higher accuracy than GA. This can be attributed to the upper tails of the estimated distributions being close. A change in the tolerance of the detection algorithm could expose this further.

In a *real* network scenario, it would be necessary to tune and adapt the methods based upon some empirical observations and knowledge of the network under test. Whilst the methods would work without changes, to extract the best performance, ie the highest bottleneck detection accuracy, such changes would be necessary. The computational complexity would

remain constant with regards to the movement of the bottleneck link but may vary as the methods are adapted.

6.7 Conclusion

In this chapter two methods for improving the tomography process were examined. The first improvement method considered using a more flexible distribution, the Pearson type-1 distribution to more accurately model the delay distribution both on its own (in a similar manner to single Gaussian distribution in GA) and as the output stage in the composite P-MOM method which combined the estimation stage of MOM with a parametric output stage. Mixed results were obtained. It was seen that PRS was more accurate than GA for scenarios where the separation was small but the performance rolled off as the separation increased to the effect that GA outperformed it. The P-MOM method provided a reduced complexity alternative to MOM with similar average response although with a higher variance and a lower robustness to scaling.

The second improvement method considered the possibility of improving the detection process by using the Kullback-Leibler divergence as a method to find the link which is *most different* to the others. The KLD detector worked in situations where the bottleneck link distribution is well separated from the distributions of other links and removes the need for *a-priori* knowledge of δ which is useful in situations where no prior knowledge of the network under test is available.

Chapter 7
Applied delay estimation

This chapter departs from the tomography related work of the previous three chapters to consider the use of delay estimation as part of a method of tracking users in a privacy preserving network. This scenario is similar in some respects to that of the tomography scenario wherein there is access to data only at the end-points of a network and the need is to estimate a delay distribution. The main departure from the tomography scenario is that there is no longer access to the routing information and the goal is to determine this information from the data available.

An anonymising network such as the onion router (Tor) [8] [9] [85] [86] is designed primarily to give privacy to users browsing the Internet who wish to hide their activity from observers. Danezis [7] proposed a method for tracking such users and presented some simulation results which allowed him to determine the route and probable end-point of the network by correlating a traffic pattern between entry and exit nodes. Results presented here confirm Danezis's work using a real Tor network with real data.

Work towards a similar goal has been performed by Murdoch and Danezis [87] which assumes a corrupted node in the Tor network is available for use by the attacker. This method is active in its approach but has the advantage of being able to operate without complete knowledge of the network. The correlation (template in their nomenclature) function employed is similar to that used in this chapter.

Other works relating to traffic analysis attacks of privacy preserving networks [88] which are similar in nature have been published by Zhu *et al*. [89] and Levine *et al*. [90]; Syverson *et al*. focus directly on Tor in [91].

Introduction

This chapter is divided into eight sections. The first section (7.1) gives a brief introduction to the Tor network and its basic operation. The second section (7.2) discusses the main algorithm

explored in this chapter. The third section (7.3) details the test Tor network used in the following work. The fourth section (7.4) describes the generation of the dataset used to test the algorithm. In the fifth section (7.5) some of the assumptions made by Danezis in this chapter are discussed. Results from using Danezis algorithm are shown in the sixth section (7.6) with comments on how the algorithm degrades in the presence of noise in section (7.7).

7.1 The onion router

Tor is a network protocol designed to provide anonymity to users by encrypting data between the end-points of a route. It is generally used for Internet communications and operates on a large number of machines, mostly those of individual users with little centralised control. Tor is an overlay protocol; it uses TCP and IP to handle data transport, delivery and routing. The small volume of centralised control that exists in a Tor network (including the large default network running over the Internet) comes from the central directory servers. These maintain the *state* of the network and collect statistics and data such as which nodes are suitable for use as exit nodes, their uptime, port routing and bandwidth restrictions imposed by the node operators. This information allows Tor to determine a choice of route for a specific user's data. Traffic to and from a directory server uses a different port to that of the normal Tor traffic and as such it can be easily separated from any measurement of the data traffic.

The other types of node in use in a common Tor network are: exit nodes - which send traffic un-encrypted to its final destination, entry nodes - which accept un-encrypted traffic, encrypt and forward it into the Tor network and routers - which forward traffic between other Tor router nodes. The entry and exit nodes are generally the end points for any Tor communication. There are a large number of possible configurations where Tor is used for Internet communication but it is most common for each user node to be an entry, router and exit node; there are no problems in combining these roles.

In a typical Tor usage scenario, the user configures his browser to route traffic to the destination (i.e., a desired web-page) via Tor. This user then becomes a node in the default Internet Tor network and his traffic is routed to the node nearest (i.e. with best connection to) the desired web-server. The security of the traffic is maintained between the user and the exit node by Tor. Traffic between the exit node and the destination may be observed if it is un-encrypted, which it is unlikely to be. A graphical description of the data flow in a typical usage scenario is shown

in Figure 7.1.

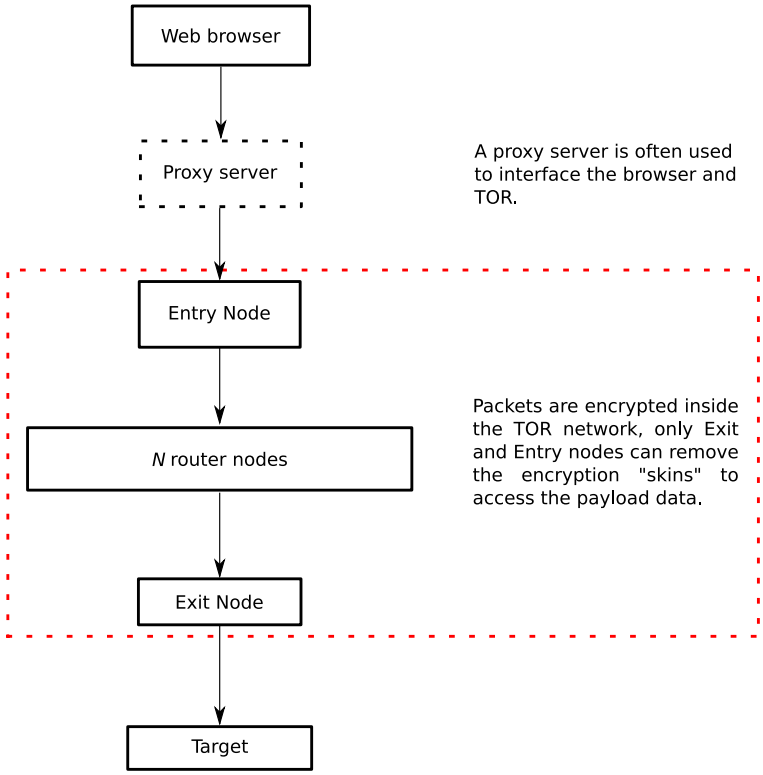


Figure 7.1: *Data flow in a Tor network from web-browser to web-server for a typical usage scenario. The Tor entry node is most likely the same machine on which the web browser is running although it does not have to be.*

7.2 Danezis's algorithm

In [7], Danezis proposes a method for attacking privacy preserving networks such as Tor based on the correlation of input and output packet streams. He assumes that such a network can be modelled as a delay mixing model (explored more in [92]) which adds delay to incoming packets in a predictable manner before ejecting them. He indicates that it is possible to deduce the exit node by calculating the correlation between the input stream and a number of possible output streams and selecting the exit node whose output stream has the highest correlation.

More specifically, what Danezis's algorithm states is: two time-series can be estimated, with lengths equal to that of the observed output streams and where each estimate is composed of the input stream (which can be thought of as a time-series), delayed by some function, added to

a uniformly distributed background traffic stream (which models network traffic which arises independently of the presence of the input stream). The background stream rate is adjusted such that it is equal to the rate of the observed stream. If one was to consider these estimated series only at the instances corresponding to observations of packets in the true output series then a more accurate estimate would be made. Division of the estimated series will indicate which output stream most likely contains the input stream.

The formula given by Danezis to estimate any output traffic distribution is shown in equation (7.1) with an explanation of the parameters used given in Table 7.1. One element which is not generally clear is the uniform distribution $U(t)$ and how it is computed: $U(t)$ is the uniform distribution in the interval $[0, T]$ and as such it conforms to equation (7.2). In any practical implementation, t will be discrete so the integral may be replaced with a sum.

$$C_X(t) = \frac{\lambda_f (d * f)(t) + (\lambda_X - \lambda_f) U(t)}{\lambda_X} \tag{7.1}$$

$$\int_0^T U(t) = 1 \tag{7.2}$$

Parameter	Meaning
λ_X	The rate of packets in the interval in question exiting node X
λ_f	The rate of packets in the interval in question in the input stream $f(t)$
$U(t)$	The discrete uniform distribution in the interval in question
u	The value of $U(t)$ at any t
$d(x)$	A function describing the delay mix of the network
$f(t)$	The input signal (stream)
$C_X(t)$	An estimate of the number of packets in the estimated output stream at t
t	The time index
$(d * f)(t)$	The convolution of the input signal with the delay mix function

Table 7.1: *Parameters used in Danezis's formulae.*

$d(x)$ is a function which represents the input/output relationship of the network in terms of packet delay, a temporal transfer function for packets. Given that this function can change over time as the network changes or as Tor changes its routing it is more practical to estimate it empirically using some training data. If access to individual nodes is possible then one can make the assumption that a given number of packets travelling on one link will have a delay distribution similar to packets on any other link, but with a change in scale. Use of a flexible

estimator for $d(x)$ such as a GMM is therefore appropriate.

Once an estimated distribution (C_X) has been computed for each output stream then they may be compared to determine which one is most likely to contain the input stream. If C_Y denotes a second output stream emanating from node Y, $X_{i=1...n}$ denotes the set of times that packets are observed at node X, $Y_{j=1...m}$ denotes the set of times packets are observed at node Y, H_0 denotes the hypothesis that the input stream in contained in the output stream from node X and H_1 denotes the hypothesis the input stream is contained in the output stream from node Y then it is possible to calculate the likelihood ratio of the two hypothesis as shown in equation (7.3).

$$\frac{\mathcal{L}(H_0|X_i,Y_j)}{\mathcal{L}(H_1|X_i,Y_j)} = \frac{\prod_{i=1}^{n} C_X(X_i) \prod_{j=1}^{m} u}{\prod_{i=1}^{n} u \prod_{j=1}^{m} C_Y(Y_j)} > 1 \tag{7.3}$$

Equation (7.3) is cumbersome to compute numerically and leads to large values which suffer from rounding when implemented in an environment such as MATLAB. However, it can be manipulated into a log-likelihood form as shown in equation (7.4) which reduces the scope of the possible values and so this form is used instead.

$$\log \mathcal{L}_{H_0/H_1} = \sum_{i=1}^{n} \log C_X(X_i) - \sum_{j=1}^{m} \log C_Y(Y_i) + (m-n) \log u > 0 \tag{7.4}$$

7.3 The Tor test network

In the work presented in this chapter a small Tor network over which full control and monitoring capabilities are available[1] is used. There are four nodes in the network: three are directory servers, routers and exit nodes (nodes B, C & D) whilst the remaining one is solely an entry node (node A). The network is restricted such that the entry node cannot be the exit node and therefore the exit must be one of nodes B/C/D. The exit node is fixed to be node B. This allows it to be used as a ground truth against which any estimates of the exit node network can be evaluated. The network topology is shown in Figure 7.2. It must be noted that this network is smaller than would be optimal for Tor. It is normal to have at least 3 hops in a Tor route, the minimum necessary to all each node to have deniability (through ignorance) about the whole

[1]This implies that is it possible to inspect any of the traffic flowing into or out of the nodes and that it is possible to configure Tor in any manner desired.

route. In the work performed in this chapter, this is does not impact on the work because it is only the exit node from the network which is being examined, not any internal connections or routing.

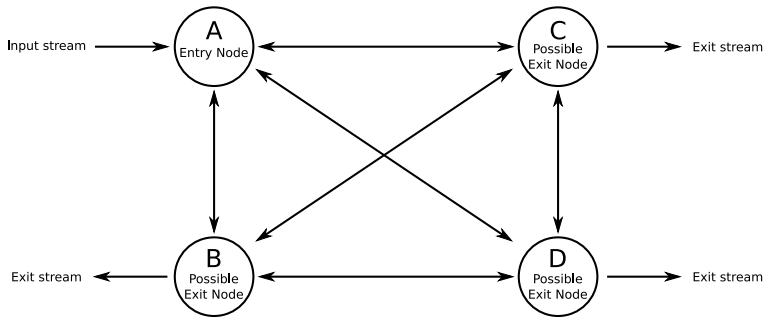


Figure 7.2: *The Tor test network.*

7.4 Generation of dataset

The dataset was recorded using `tcpdump` [93] [94] on the network described in Section 7.3 for a period of twelve hours beginning at approximately 1700 hours to minimise the effect of any user traffic on the dataset. The dataset was split into twenty four contiguous segments of thirty minutes and for each segment there were three separate traces: ALL - which recorded all packets, NOR - which recorded any packets not generated or used by Tor and OR - which recorded only packets generated or used by Tor. Also recorded was the traffic entering the transparent proxy server (privoxy [95]) used to interface the web-browser and Tor, `privoxy`. Cutting the dataset into a number of segments enables averaging of results across segments in order to reduce the impact of any unusual network circumstances.

Network traffic was generated by directing a web-browser (Konqueror) to access the home-page of the BBC news website (http://news.bbc.co.uk) at timed intervals. The intervals are on the order of seconds and randomly generated as the result of selecting a random number from a Poisson distribution with parameter equal to 30. Using a Poisson distribution removes the periodicity which would be encountered with a uniform process but retains some element of regularisation. Algorithm 8 shows the traffic generation method.

Filtering was performed to isolate packets serving different functions by port number and included removal of any SSH packets (port 22) which were part of the recording or monitoring

Algorithm 8 Network traffic generation

1: **loop**
2: Generate *interval* using Poisson-based random number generator with parameter equal to 30.
3: Start web browser and access web page via Tor network.
4: Wait for *interval* seconds.
5: Kill web browser.
6: **end loop**

processes (such as status indicators).

The traces were processed using a range of AWK scripts to extract packet delays, time series and trace start times. By inserting the time the trace began into the trace file it is possible to manage a situation where there is no traffic in the early part of a trace which could otherwise cause a trace to appear to be less than 1800 seconds in duration and packets to appear skewed relative to packets in parallel traces.

Once processed, traces were loaded into MATLAB for further processing. The data were first scaled by the trace start time i.e., subtracting from all elements in the series the lowest value before using the `histc` function to convert from a series of timestamps to a binned representation of the data. The scaling allows the binned respresentation to be computed with an arbitrary resolution for any segment without having to change the resolution of the whole dataset, i.e. it makes the segments independent of the time they were recorded.

7.4.1 TCP control packets

Further to the above discussion in which all packets were sampled for each trace, it was possible to exclude many TCP control packets. These are defined here to be the packets which encapsulate no Tor data, but are part of normal TCP operation; examples include SYN and FIN packets. Inclusion of the control packets increases the observed data rate. Where only the number of packets per second and not the type is considered, the dataset can appear larger. This is useful where a large number of packets is needed to ensure that the signal is not represented in a sparse manner, which may give a poor result.

7.5 Development of results

Before examining results for a whole datasets, presented below are some example results which show some of the assumptions made and elaborate upon the development of the method. A thirty minute segment of data from the dataset is used in this example.

In his paper, Danezis gives no source code and little implementation detail so the following assumptions are made:

1. It is assumed that the algorithm is functional when implemented in a discrete form. It is originally presented in a continuous form which does not lend itself to easy implementation. Variables such as C_X are treated as discrete vectors by applying a binning process to the continuous time series data which arises from measurement. This implies that C_X depicts packet counts over the segment of interest with t being the bin index. Similarly, $f(t)$, $d(x)$ and the other estimated series (C_Y etc.) are binned with the same resolution and are thus X_i and Y_j are intrinsically discretized. Perhaps more intuitively C_X, $f(t)$, X_i and Y_j can be thought of as packet counts per unit of time where t indexes the time interval and where the temporal resolution is equal for all variables including $d(x)$.

2. It is assumed that the algorithm is robust to changes in the scaling of t such that it is possible to vary the resolution of t and therefore the bin width of C_X with the caveat that the temporal resolution must be consistent across all variables.

3. It is assumed that Tor traffic can be separated from other traffic at any node. The Tor traffic which flows from one exit node to any other Tor nodes (to pass ACKs back upstream, for example) will be highly correlated with the exit stream and may disrupt the algorithm.

4. It is assumed there is enough data to model the delay function, $d(x)$, using a GMM of three elements; should it be found that a three element mix has redundancy, the order is reduced to two or one. Tor incoming packet delays are measured at all exit nodes for packets originating at other Tor nodes and it is assumed that the statistical distribution of these packets is representative of the delay distributions experienced by packets exiting the Tor network.

5. It is assumed λ_f (and subsequently, λ_X, λ_Y, etc.) is computed as the mean number of

packets per step (one step is the elapsed time between time index t and time index $t+1$) over the interval $[0, T]$. Thus, if there are 2000 packets in an interval of 25 seconds with a step size of 0.1 s then the rate is $2000/(25/0.1) = 8$ packets per time step.

To begin with, the input stream (Figure 7.3) is observed in isolation. The traffic appears to be near-periodic but with some variation which is expected given the generation method used.

Next, the output streams from each of three possible exit nodes is observed (Figure 7.4 (a)). Any packets travelling towards the target web-server have been filtered out to show the background traffic at each exit node with a temporal resolution of one second. Node C is observed to send out a periodic stream of packets; as there is no Tor traffic or exit-stream traffic then it can be assumed this is either due to a user process running on this machine or, more likely, a network operation (backup, file handling etc). Node B sends out more frequent traffic than node C and the lack of periodicity would indicate a user process (web browsing, email etc). Node D has little outgoing traffic perhaps because there were no active users during the time the traffic capture was in operation.

Finally, the outputs from each of the possible exit nodes are observed when the input stream is applied to the network (Figure 7.4 (b)). There is a large rise in the volume of traffic emanating from Node B with a near-periodic pattern which appears to be similar to the input stream giving the first indication this may be the exit node.

7.6 Results

Danezis compares the two distributions in his short example using a simple ratio test. Whilst it would be possible to scale this using a tree-search type of algorithm, it makes sense to use some form of ratio combining algorithm. Define the ratio A to be the ratio of the estimated distributions for nodes B and C; B to be the ratio for nodes C and D and C to be the ratio for nodes B and D. The estimated distributions are the instantiations of equation (7.4) with the relevant data.

For the segment discussed previously, Figure 7.5(a) shows the ratios and Figure 7.5(b) shows the log-likelihood ratios.

Thus, it is possible to define the following formulae:

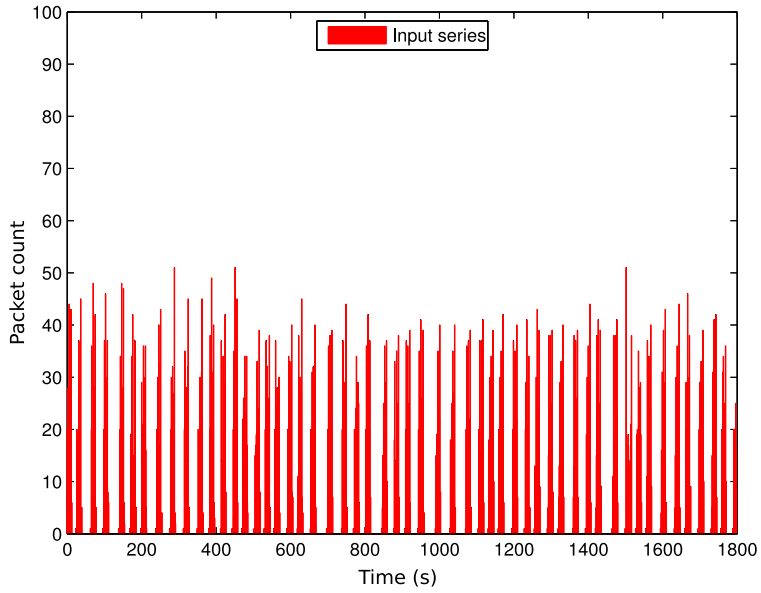


Figure 7.3: *The input stream in isolation.*

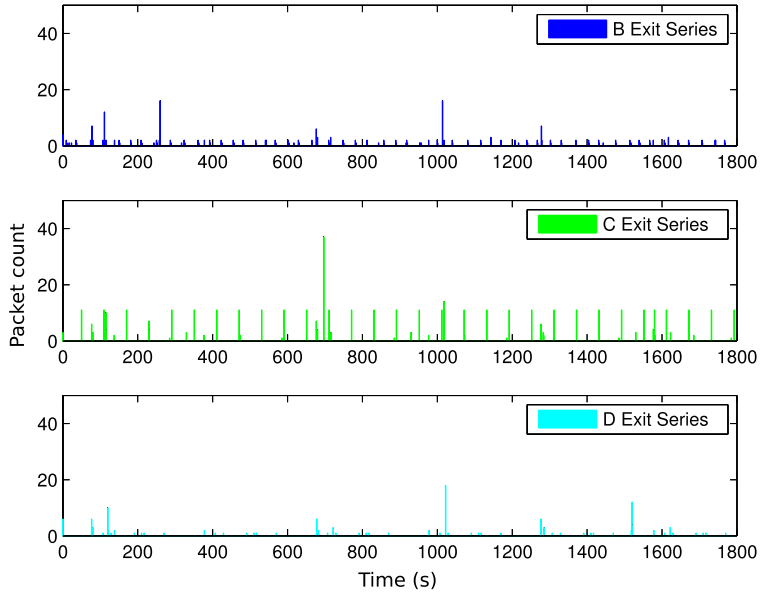
$$P_B = A + C \tag{7.5}$$

$$P_C = -A + B \tag{7.6}$$

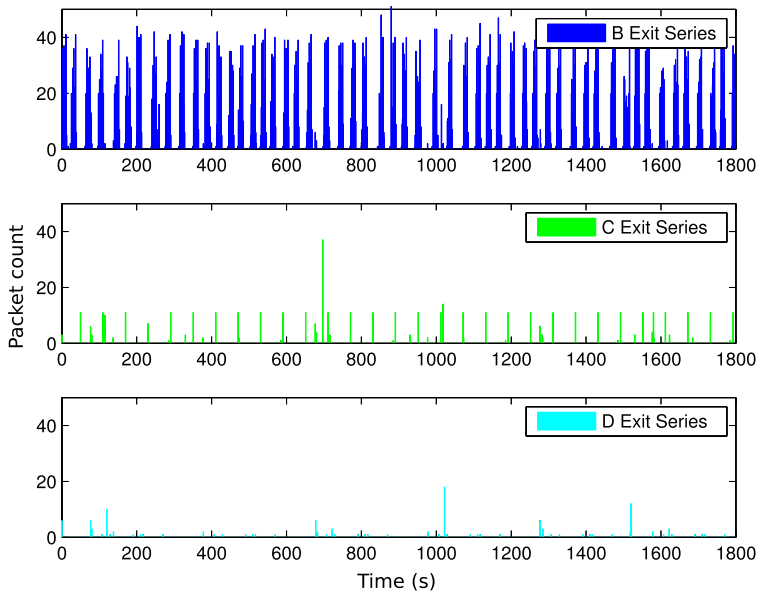
$$P_D = -B + -C \tag{7.7}$$

By selecting $\max\left[P_B, P_C, P_D\right]$ the most likely exit node for the data in question can be deduced; the difference between the selected node and the others gives some measure of confidence in the decision.

The results for this dataset with the TCP packets excluded are shown in Table 7.2. It can be seen that the algorithm has correctly identified node B as the exit node in each of the twenty-four segments. However note that the delay function $d(x)$ was estimated to be 1 at the first time index and 0 elsewhere. This can be interpreted as an indication that the model for the delay is at a resolution smaller than the resolution at which the algorithm is being run. A simple check can demonstrate this. Delays on the order of milliseconds are observed but the algorithm is run at a resolution of seconds hence the binned approximation to the delay function being 1 in the first bin and 0 elsewhere.

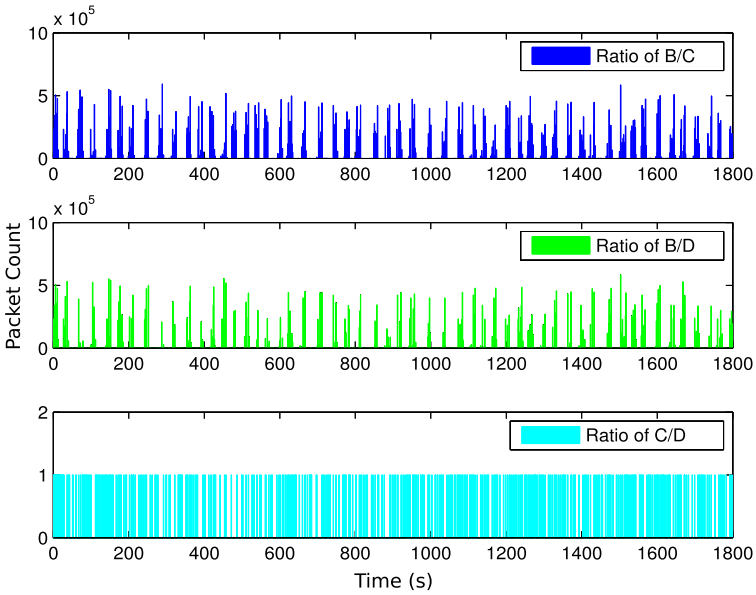


(a) Background traffic at each node.

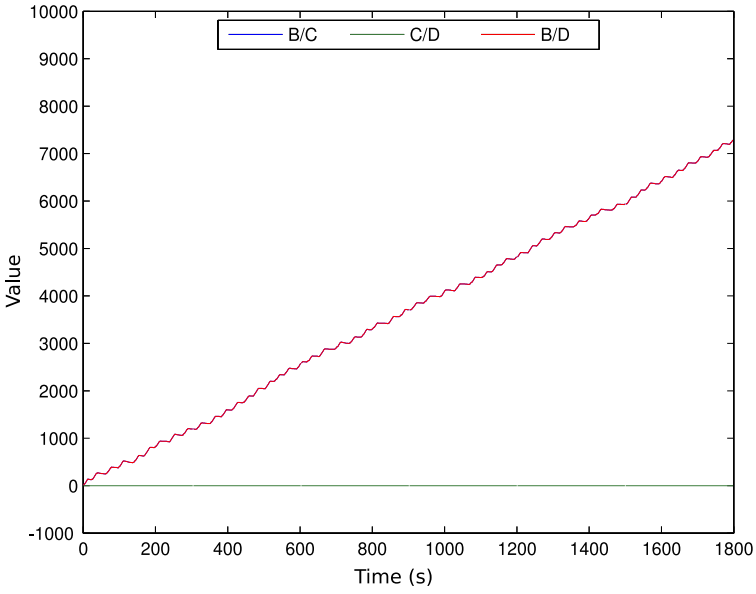


(b) Background traffic and network output at each node.

Figure 7.4: *The output streams from the three possible exit nodes showing: (a) - no input being applied to the network, i.e. the background traffic and (b) - the result when the input stream is applied.*



(a) Standard ratio



(b) Log-likelihood ratio

Figure 7.5: *(a) The three ratios computed at each step in t using equation (7.3) and (b) the log-likelihood ratio for the interval $[0, t]$ computed using equation (7.4).*

Resolution (s)	Node B	Node C	Node D
1	24	0	0
0.1	24	0	0
0.01	24	0	0

Table 7.2: *Results for the dataset with without TCP. These indicate the number of times that each node is chosen as most likely exit node and are scored out of 24 - the total number of segments for this dataset.*

Resolution (s)	Node B	Node C	Node D
1	24	0	0
0.1	24	0	0
0.01	24	0	0

Table 7.3: *Results for the dataset with TCP. These indicate the number of times that each node is chosen as most likely exit node and are scored out of 24 - the total number of segments for this dataset.*

What the results in Table 7.2 and Table 7.3 show is that the presence (or lack) of the TCP packets has no noticeable effect on the performance of the algorithm at these resolutions.

7.7 Degradation of algorithm

The results presented so far show the algorithm performing well with a low volume of noise in the system. The motivation for this section is to determine circumstances where the algorithm's performance degrades.

The results shown below are constructed using the dataset and algorithm used previously but with background-traffic added to the binned exit streams in the form of an equal number of packets per bin. Thus, when it is said that a background-traffic of 10 is added to an exit stream, each bin in the binned representation of the exit stream had 10 packet counts added to it. The metric of measurement is the number of times each node is chosen as the most probable exit node with node B being known, *a-priori*, to be the correct choice. Rs_X denotes the number of times (from a total of 24) node X is chosen as the probable exit node; R_X denotes the rate of the exit stream emanating from node X (in packets/second) and N_X denotes the background-traffic added to the exit stream of node X (in packets/bin).

Choice of node as exit (score/24)			Rate of exit stream (packets/second)			Added background traffic (packets/bin)		
Rs_B	Rs_C	Rs_D	R_B	R_C	R_D	N_B	N_C	N_D
24	0	0	6.80	0.27	0.07	0	0	0
24	0	0	6.80	7.27	7.07	0	7	7
17	6	1	6.80	8.27	8.07	0	8	8
13	10	1	6.80	9.27	9.07	0	9	9
12	11	1	6.80	10.27	10.07	0	10	10
9	14	1	6.80	16.27	16.07	0	16	16
10	1	13	6.80	19.27	19.07	0	19	19
20	0	4	6.80	100.27	100.07	0	100	100
24	0	0	6.80	200.27	200.07	0	200	200

Table 7.4: *Accuracy results for exit node estimation with varying background-traffic levels.*

Tables 7.4 and 7.5 show the exit-node estimation accuracy results obtained by varying the noise added to each of the output streams.

Table 7.4 shows three distinct regions of operation. The first region (row 1) is where R_C and R_D are lower than that of the input stream: the exit stream cannot contain the input stream and node B is selected as most probable exit node in each instance. In the second region (rows 2 & 3) noise is added to both C_C and C_D until R_C and R_D exceed R_B, i.e. when a noise of 8 is added. The accuracy begins to fall because the estimated distributions C_C and C_D have at least one packet in each bin and C_B does not. This causes R_B to be lower than R_C and R_D. This continues as the level of noise is increased until a noise of 19 is reached. In the third region (rows 4 - 9), the scaling elements of C_C and and C_D are such that the value of $\sum \log C_C$ and $\sum \log C_D$ are less than that of $\sum \log C_B$ despite C_B not having a packet in each bin. The upward trend in accuracy continues as N_C and N_D are increased due to the scaling factor increasing, resulting in full accuracy with large noise, ie 200.

Table 7.5 shows a similar result. In the first region (rows 1 & 2), R_C and R_D are less than that of the input stream and nodes C and D cannot be chosen as discussed above, regardless of N_B. In the second region (rows 3 - 6) a fixed value of noise is added to C_C and C_D and the noise added to C_B gradually increased. When a noise of 1 is added to C_B degradation occurs because each of the three vectors will have $\geq$ 1 packet each bin and C and D win due to the

Choice of node as exit (score/24)			Rate of exit stream (packets/second)			Added background traffic (packets/bin)		
Rs_B	Rs_C	Rs_D	R_B	R_C	R_D	N_B	N_C	N_D
24	0	0	6.80	0.27	0.07	0	0	0
24	0	0	26.80	0.27	0.07	20	0	0
12	11	1	6.80	10.27	10.07	0	10	10
3	20	1	7.80	10.27	10.07	1	10	10
5	19	1	9.80	10.27	10.07	3	10	10
24	0	0	10.80	10.27	10.07	4	10	10
2	1	21	10.80	20.27	20.07	4	20	20
24	0	0	10.80	100.27	100.07	4	100	100

Table 7.5: *Further accuracy results for exit node estimation with varying background-traffic levels.*

weight of numbers. The accuracy increases as N_B is increased until R_B exceeds R_C and R_D where C_B will have heavier weight. In the third region (rows 7 & 8), when N_C and N_D are increased to 20 a poor accuracy is observed. However, if N_B is large, ie 100, the accuracy is good as the scaling effect reduces the weight of C_C and C_D relative to C_B.

7.8 Conclusion

In this chapter, an alternative use of delay estimation methods has been explored as part of an algorithm to track users using a privacy preserving network such as Tor. An algorithm first proposed by Danezis was implemented and tested using real data captured from a live Tor network. It was found that some of the assumptions Danezis made were invalid (particularly the length of his delay function) and that it was necessary to make some additional assumptions to achieve a workable implementation.

In terms of future work, it would be useful to run a test where there was a significant volume of cross-traffic inside the Tor network. This would arise from different entry and exit combinations as happens in a *real* Tor network and be used to test the robustness of the algorithm. A scenario with very little non input-stream related Tor traffic was tested here. It would be beneficial to know how clear the input stream must be, with regards to the other Tor traffic, for the algorithm to correctly identify a link carrying, or an exit node emitting, a particular input-stream.

Chapter 8
Summary and conclusions

The aim of this thesis was to examine delay estimation methods in computer networks. Chapters 4 to 6 concentrated on the scenario of network tomography as a setting for the work and examined the hypotheses that:

- The method of moments (MOM) could be bettered as a method for network tomography.
- The single Gaussian distribution (GA) could be used despite misgivings over identifiability.
- The Kullback-Leibler divergence (KLD) could be used as a detector in suitable scenarios.
- Improvements to the GA could be devised by using either Gaussian mixture models (GMMs) or other univariate distributions.

The final technical chapter saw delay estimation used in the context of a delay model for a user tracking algorithm in a privacy preserving network. Results from the tomography work proved valid in this context with regards the accuracy of GMMs and their suitability for use as delay estimators.

8.1 Achievements and contributions

The first technical chapter, Chapter 4, described the comparison between the GA and the MOM methods; both were previously published but had not received a *side-by-side*, objective performance comparison. Using three metrics, the results showed that GA was a computationally efficient method with a performance which can exceed that of MOM in a scenario where the delay separation between the bottleneck and next-worst links is small. It was observed that GA was more robust to a reduction in probe packet rate than MOM with the accuracy of GA degrading at a rate slower than that of MOM. GA's primary disadvantage is sensitivity to the choice of the delay threshold (δ), poor choice can affect an accuracy change

of up to 60% in the most extreme case. The impact of the problem is reduced as separation increases and it may be concluded that the blame most likely lies with the CDFmax detection algorithm.

It was suggested in [82] that there would be problems with identifiability using the GA method but this hypothesis was found to be false under the assumption that full knowledge of the routing matrix (RM) is available. Results also showed that MOM was a robust method and suitable for use where no *a-priori* knowledge of δ was available. This was seen to come at a cost of high computational complexity even when considering a limited (compared to the published algorithm) number of moments.

In Chapter 5 GA was developed into a more flexible method by using a GMM in order to increase the accuracy of the delay distribution measurement and estimation. There were two challenges to this approach: finding the optimal number of elements in the GMM and re-combination of inferred elements into a usable and valid distribution function. Two algorithms were presented to address the issues: the Sum-of-Gaussians A (SOGA) which fits the measurements with a GMM and *then* infers the link distribution and the Sum-of-Gaussians B (SOGB) which infers link data direct from raw measurements and fits a GMM to these. It was found that SOGA has an accuracy similar to GA except in scenarios with small separation where it outperforms GA and, by extension, MOM. With respect to accuracy, SOGB performs poorly but does so at a much reduced computational complexity; better (i.e. less demanding) than MOM but not as good as GA; by comparison, SOGA is the worst of the four with a computation time for one segment of data more than 70 times that of GA.

Chapter 6 presented two ideas: one, an improved detection algorithm based upon the Kullback-Leibler divergence and two, the use of the Pearson type-1 distribution in a fully parametric and a hybrid-parametric method. The KLD-based detection algorithm results suggested its viability but only when the separation was sufficiently large. Subsequently, the goal of finding a detection algorithm with no dependence on the delay threshold was achieved but the necessity of a large separation implied that it would not be feasible in anything other than an anomaly detection scenario. The Pearson type-1 distribution (PRS) method demonstrated that it was possible to use this flexible distribution to model delay measurement distributions but a valid distribution is not always obtained from the inference process and often requires correction. Its performance exceeded GA in a scenario with a small separation but in other scenarios its accuracy was lower. Computational complexity was

worse (i.e. it was more computationally demanding) than GA but better (i.e. less demanding) than MOM. A potential niche for use is in a small separation scenario where the complexity of SOGA is too high but more accuracy than GA is required. The hybrid method Pearson method-of-moments (P-MOM) which was formed by feeding the estimated moments of MOM into the output cumulative distribution function (CDF) of a Pearson type-1 distribution was examined and found to offer an accuracy which was similar to MOM (albeit with some variance) at a reduced complexity - between 5 and 10 times fewer operations. This then adds another option where Independence of the method from δ is required and computational complexity is limited.

In Chapter 7 an implementation was presented, using real-world data, of the algorithm first proposed by Danezis [7]. It was shown that it is necessary make some assumptions about the nature of the data before implementation and that it is possible to discretize the function with an arbitrary temporal resolution. It was demonstrated that this algorithm was successful in identifying the exit node in a network with three possible choices. The algorithm was adapted to perform a similar task in searching the dataset to see if it was possible to reconstruct a route for the data from exit node to entry node. Whilst a complete route was not constructed, the results presented indicated the method was viable in practice.

8.2 Future work

This thesis introduced seven methods for network tomography and tested them using ns-2 simulations. As an extension of the work it would be relevant therefore to test the methods in a real network scenario to generate an authoritative result as to the relative accuracy of each method, demonstrate the sensitivity to δ and by extension the typical values of δ found in real networks, substantiate the conclusions about robustness and scaling (a real network is likely to be greater than 10 nodes in size) and confirm the extent to which the measures of complexity correlate to processing time. It would also give an insight into the importance of the parameters used in simulation: the degree to which the traffic models match real models, how probe traffic affects network characteristics, how true the assumptions about the Poisson arrivals see time average (PASTA) and periodic probing hypotheses are and whether or not topologies remain constant over typical periods of observation (and indeed what these periods are).

It would also be relevant to consider the measurement and inference of network characteristics

other than packet delay. The review of literature examined algorithms which sought to infer available bandwidth (ABW), loss-rate and jitter as well as delay. Initially it may seem trivial to change the characteristic being measured but it may be the case that distributions for loss-rate, for example, are not as well modelled by a Gaussian distribution as delay and as a consequence the methods using GMMs or the PRS may be more appropriate, leading to a different opinion about the trade-off between computational complexity and accuracy.

One further application of delay inference and measurement would be as the input to a control system. There are quality of service (QOS) schemes which use delay as a metric of interest and some routing schemes which use delay to determine routes at the network layer. These types of application require a high speed of computation and robustness to changes in the network environment. Thus, they may require changes to the tomography algorithms. Instead of detecting which link is most likely to be the bottleneck, what is required are estimates of the absolute value of the delay. In effect, this is the version of tomography used in this thesis but without a detection stage.

Appendix A
Machine specifications

The table below shows the specifications of the machine used to compute the run times for the various methods in Chapters 4, 5 and 6.

Parameter	**Value**
Machine architecture	x86_64
Number of CPUs	2
Number of cores per CPU	1
CPU Speed	3192 MHz
Voulme of RAM	2058072 KB
Operating system	Scientific Linux SL release 5.3 (Boron)
Kernel version	2.6.18-128.7.1.el5
MATLAB Version	2008a

Table A.1: *A summary of key machine specifications used to compute method run times in this thesis.*

Appendix B
Publications

The following papers, based on the work in this thesis have been published, and are reproduced in this appendix.

N. Johnson, J.S. Thompson, S. McLaughlin and F.J. Garcia, *Network tomography, Delay estimation & Bottleneck link discovery*, 2008 IAPR workshop on Cognitive Information Processing, June 9th - 10th, Santorini, Greece

N. Johnson, J.S. Thompson, S. McLaughlin and F.J. Garcia, *A Comparison of Delay Estimation & Bottleneck-link Detection Methods for Network Tomography*, 8th IMA Mathematics in Signal Processing Conference, December 16th - 18th, Cirencester, UK

N. Johnson, J.S. Thompson, S. McLaughlin and F.J. Garcia, *A comparison of some bottleneck-link detection methods for network tomography*, 17th European Signal Processing Conference, August 24th - 28th, Glasgow, UK

NETWORK TOMOGRAPHY, DELAY ESTIMATION & BOTTLENECK LINK DISCOVERY

Nick Johnson[*]*, John Thompson & Steve McLaughlin*

Institute for Digital Communications
School of Engineering and Electronics
The University of Edinburgh
Edinburgh, EH9 3JL
Nick.Johnson@ed.ac.uk

Francisco J. Garcia

Agilent Laboratories, Scotland
South Queensferry
Edinburgh
EH30 9TG
Frankie_garcia@agilent.com

ABSTRACT

An important issue in the measurement of networks is the ability to infer characteristics of internal network links from measurements made on end-to-end paths. It may be impractical in terms of equipment, time or cost to monitor each individual link but it is often feasible to monitor a number of existing paths. Provided there is enough traffic flowing through enough different paths then it is possible to estimate some characteristics of each link. In this paper we compare two methods for estimating the end-to-end delay distributions, one based on the method-of-moments and the other on a Gaussian approximation. This information can then be used to compute packet delay on any link in a network and then detect which link has the highest latency. This procedure is often termed bottleneck link discovery.

Index Terms— Network Tomography, Delay Distribution Estimation, Network Inference, Estimator Comparison

1. INTRODUCTION

The term network tomography is first used by Vardi in [1]. In [2], [3] & [4] the term network tomography is used to define an approach to infering network characteristics from a limited subset of measurements made in a wired network. We consider here a specific type of network tomogrpahy, the problem of estimating link-level characteristics from path-level measurements. This approach is used in [5] [6] [7] [8] because it is often impractical and inefficient to measure all internal links in a network. It is possible to infer from a set of measurements taken over a selected set of paths (where a path is a combination of links) the likely delay on each link. From these estimates a method of detecting the link with the highest delay can be used to detect a bottleneck link. Once detected, the link can be modified to reduce the delay or the network routing can be changed to lower the volume of traffic on that link.

There are two methods of gathering an estimate of the delay mentioned in the above papers. One method (used in [2]) attempts to estimate the Cumulant Generating Function (CGF) of the delay using measurements of the delay of probe packets. Another (used in [4]) is to assume an a-priori delay distribution then estimate the parameters of this distribution from the delay of probe packets. Our contribution is to use the ns2 simulator [9] to compare the two methods' ability to correctly identify the bottleneck link. Performance with a reduced number of probe packets and the computational complexity of both methods will also be studied.

The remainder of this paper is organised as follows. In Section 2 we review the methods used in [2] and [4]. In Section 3 we present results from simulations comparing both methods and an estimate of computational efficiency. Finally, in Section 4 we give some conclusions and provide pointers to further work.

[*]Funding for this work is provided by Agilent Technologies via a PhD Fellowship Award

2. SYSTEM MODEL & ESTIMATORS

2.1. GENERAL MODEL

A network of routers can be modelled as a set of connected links with the connections specified in a routing matrix. We define a path to be a connected set of two or more links from the total set of links L. An estimate of the distribution of delays is formed from the delays of unicast probe packets on various paths whose total number is P (See Fig 1, originally from [3] where $P = 5$ & $L = 4$ for illustration). The vector of delay observations on path i is represented by $Y_i, i = 1...P$ with the routing matrix represented by H which is of size $P \times L$. The objective is to find an estimate of the distribution of delays on each link, represented by $X_j, j = 1...L$. Equation 1 shows the linear relationship between these three quantities:

$$Y = HX \tag{1}$$

For the network shown in Fig 1 we define H as:

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (2)$$

Provided H is full rank then it is possible to use a least-squares (LS) algorithm to recover X:

$$X = H^{-1}Y \quad (3)$$

We define H^{-1} as the pseudo-inverse of H as in [3]:

$$H^{-1} = (H^T H)^{-1} H^T \quad (4)$$

This pseudo-inversion must be performed each time the topology changes to ensure the correct weights in the LS algorithm. For a wired scenario, as considered in this paper, is it likely to be infrequent operation.

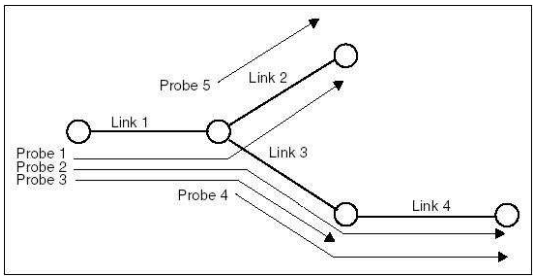


Fig. 1. Network Topology & Probe Paths

Once we have an estimate for the distribution of delay (or other network parameter) on a link it is desirable to compute the bottleneck link.Where we have delay distributions, we examine the cumulative distribution function (CDF) of all links and compare them. We seek to find the link that has the largest value which, when they are normalised, entails finding the CDF with the heaviest tail. To do this we supply a value, normally δ, which is the value at which to compare the CDFs. We choose the link with heaviest tail as our first choice bottleneck link and could continue, if necessary, to select the second, third etc. choice links. It should be noted that bottleneck link detection is not always performed and not always useful. Consider the case of a network where each link has a similar performance but with small perturbations. A bottleneck link detection method would likely identify one link as poorly performing where its performance is comparable to others.

In the remainder of this section, we will introduce the techniques under consideration.

2.2. Method of Moments

In [3] the authors estimate the CGF of the distribution of delays on each path from individual delay measurements with a method-of-moments (MoM) estimator, yielding Y. These are passed through the LS algorithm to give a CGF of the delay distributions for each link in the network.

We first construct an estimate of the CGF of path i using N measured delays denoted $Y_{ik}, k = 1...N$,

$$\widehat{M_{Y_i}}(t) = \frac{1}{N}\sum_{k=1}^{N} e^{tY_{ik}} \quad (5)$$

Then we use LS to obtain a link-level estimate of the CGF where h_{ij} is the i^{th} row and j^{th} column element of H^{-1} and hence we sum the weighted contribution from each path towards that link.

$$\widehat{K_{X_j}} = \sum_{i=1}^{P} h_{ij} \times \log(\widehat{M_{Y_i}}) \quad (6)$$

To find the link with highest delay a Chernoff upper bound is imposed on the link CGFs. The link with the highest probability, P_j of exceeding the delay threshold, δ is taken to be the bottleneck link in the network.
In [3] this is expressed as:

$$P_j = P(X_j \geq \delta) \leq e^{-t\delta} E[e^{tX_j}] \quad (7)$$

This method necessitates a-priori selection of the value of δ to be used as the delay threshold.

2.3. Gaussian Approximation

In [4] the authors suggest the CDF of delays on links could be modelled as a single Gaussian distribution.

We can estimate the mean of the delay distribution on path i as:

$$\widehat{M_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} Y_{ik} \quad (8)$$

And similarly the variance:

$$\widehat{\sigma^2_{Y_i}} = \frac{1}{N-1}\sum_{k=1}^{N} (Y_{ik} - \widehat{M_{Y_i}})^2 \quad (9)$$

We express the distribution of delays on a particular link, X_j, as a single Gaussian by using LS thus:

$$X_j = \mathcal{N}(\sum_{i=1}^{P} \widehat{M_{Y_i}} \times h_{ij}, \sum_{i=1}^{P} \widehat{\sigma^2_{Y_i}} \times |h_{ij}|^2) = \mathcal{N}(\widehat{M_{X_j}}, \widehat{\sigma^2_{X_j}}) \quad (10)$$

Note that we model the variance as a noise process so multiply by $|h_{ij}|^2$ in order to preserve the positive sign.

To find the bottleneck link we evaluate the erfc function (which normally applies to $\mathcal{N}(0,1)$ and is modified in Eqn 11) at δ for each X_j and select the link which has the highest value of P_j. Again, this necessitates a-priori selection of δ.

$$P_j = \text{erfc}(\frac{\delta - \widehat{M_{X_j}}}{\widehat{\sigma^2_{X_j}}}) \tag{11}$$

3. SIMULATION STUDY

3.1. Simulation Setup

To test both methods we use an ns2 simulation to model a wired network with unicast probe-path traffic. The topology is as shown in Fig 1 and the simulation parameters are identical for both methods.

Background traffic on each link is formed by combining a number of exponentially distributed UDP and a number of TCP traffic sources in a similar manner to that used in [2]. On each link we add a delay to each packet to force a situation where one link has higher latency than the others for both methods to detect. Aside from the added delay, other delays encountered by packets come from self-congestion due to background traffic in the form of queueing and processing time at each node. Key simulation parameters are given in Table 1.

Parameter	Value
Added Delay Link 1	100 + [10-60] ms
Added Delay Link 2	100 ms
Added Delay Link 3	80 ms
Added Delay Link 4	10 ms
Bandwidth on each link	1 Mb
Simulation Time	1000 s
Number of Paths, P	5
Number of Link, L	4
CGF Parameter, t	20
Value of δ for comparison	0.15
Number of samples, N	3000
Estimator Rate	2 Kb/s
Estimator packet size	40 Bytes
Background Traffic Link 1	800 kb UDP, 1 TCP
Background Traffic Link 2	600 kb UDP, 1 TCP
Background Traffic Link 3	900 kb UDP, 1 TCP
Background Traffic Link 4	300 kb UDP, 3 TCP

Table 1. Key Simulation Parameters

3.2. Key Results

In this section we present results showing the comparison between both estimators (the method-of-moments (MoM) and the Gaussian approxiamtion (MoG)) for different observation window sizes and estimator rates.

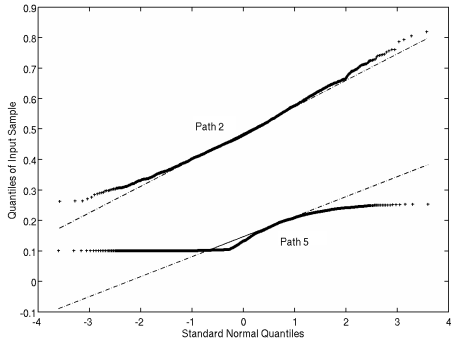


Fig. 2. QQ plot of data on paths 2 & 5

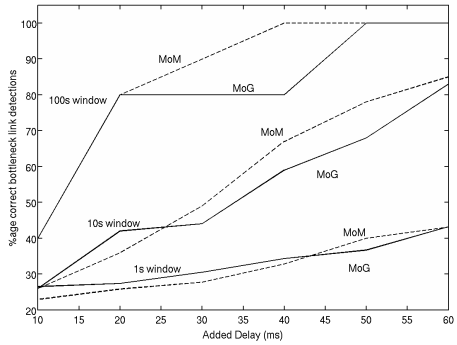


Fig. 3. Added Delay against Correct Detection Rate

To compare both methods we perform the simulation then compute the bottleneck-link for 3 scenarios:

- 1000 realizations of a 1s observation window
- 100 realizations of a 10s observation window
- 10 realizations of a 100s observation window

In Fig 2 we see a quantile-quantile plot of the delays of estimator packets on paths 2 & 5. The fit between the solid line (the data) and the dashed line (a Gaussian) indicates how well a single Gaussian models the data. On path 2, a single Gaussian distribution gives a good model of the data whereas on path 5 a single Gaussian gives a poor fit. The poor fit in the lower tail of path 5 is evident on other links and is a result of packets having a minimum delay which causes deviation from the single Gaussian model. As the fit for most paths is good in the central 4 quartiles we use the single Gaussian distribution as an estimator.

In Fig 3 we see that with the longest observation window (100s) MoM is 100% reliable for added delay greater than 40ms whereas MoG achieves 80% reliability. When the added delay is reduced to 20ms both methods converge to 80% reliability and continue to experience the same reliability value as the delay is reduced further. For a shorter observation window (10s) a similar trend can be observed; at 50ms added delay MoM achieves 78% reliability with MoG achieving 68%. As added delay is reduced, both methods convergence in performance so that MoG achives 42% reliability at 20ms compared with 36% for MoM. As added delay is reduced to 10ms, both methods exhibit similar performance. With a short observation window (1s) we observe the same trend as with longer windows but with much reduced reliability. With 50ms added delay MoG has a reliability of 37% while MoM achieves 40%. With 30ms added delay MoG performs best with 31% reliability compared to 28% with MoM. This trend continues with MoG being 27% reliable at 10ms and MoM being 23%. In this case both estimators achieve very similar performance.

From the above we note that with a long observation window and with an added delay greater than 20ms the MoM provides the most reliable method of bottleneck-link detection. If the added delay is reduced to less than 20ms and the observation window shortened to 10s or less then the MoG achieves a similar performance. Practically, this implies that in a network where delays on a link are within 20ms of each other and only a short observation period is available then using a either method would be equally likely to provide correct detection of the bottleneck link.

3.3. Estimator Rates

One consideration for both methods is the number of probe packets required to perform reliable bottleneck-link detection. Here, we consider the effect of reducing the probe traffic rate on both methods studied above; reduction of the probe rate improves efficiency by congesting the links with fewer probe packets, however, this is at the cost of accuracy.

To evaluate this trade-off we use the topology and estimation methods shown previously but adjust the simulation parameters such that Links 1, 2 & 3 have 10ms and Link 4 has 50ms added delay. We use 100 realisations of a 10s observation window to remain consistent with previous results.

From Fig 4 we see that with an estimator rate of 1Kb/s (half that used in the previous scenario) the MoM achieves a reliability of 94% with MoG achieving 92%. As the rate is reduced to 0.7Kb/s the MoM reliability is reduced to 87% whilst the MoG achieves only 64%. As the rate is further reduce to 0.5Kb/s, a quarter of the original, the reliability of the MoG falls to 49% whilst the MoM has fallen to 80%. This suggests that a MoM approach is preferable when the number of probe packets is low. This would appear consistent with Fig 3, reducing the estimator packet rate and reducing the observation window have the effect of reducing the data

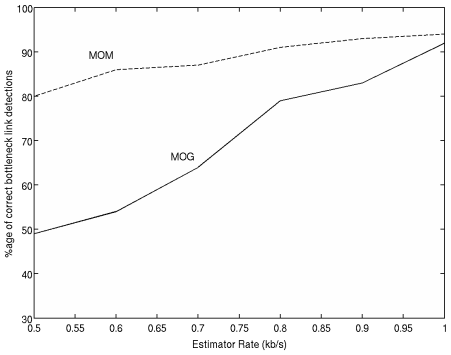


Fig. 4. Estimator Rate against Correct Detection Rate for both methods with a 10s observation window

available to the estimator which results in a lower probability of correct bottleneck link detection. The corollary also applies, the probability of correct bottleneck detection can be improved by either raising the estimator packet rate or increasing the observation window to increase the data available to the path-level estimator.

3.4. Computational Complexity

We compare the computational intensity of each method by considering the number of Multiply (MULT) and Add (ADD) operations required:

	MoM	MoG
MULT	$2NPt + Pt + PL - L$	$PN + 2P + 3PL$
ADD	$NPt - Pt + PL - L$	$2PN + 2PL - 2L$

Table 2. Formulae for number of operations required

In Table 2 we present equations for the complexity of the data processing part both methods, (ie excluding the sampling process) in terms of the simulation parameters. We see the complexity of both methods scales by the number of samples, N, but that MoM also scales by CGF parameter, t. Here, and in the scenarios described in Section 3.1, P, L & t are 5, 4 & 20 respectively. In Table 3 and Fig 5 we see the number of operations required in the scenarios previously mentioned where we observe that MoG requires an order of magnitude fewer operations than MoM. However, this comes at the expense of the reliability of bottleneck-link detection.

Fig 5 shows graphically how the number of operations quickly scales as the size of the network increases. Here we have assumed the ratio L/P has remained constant at 0.8 as in the previously defined scenarios.

	MoM	MoG
MULT	600116	15070
ADD	299916	30032

Table 3. Numerical results for number of operations required

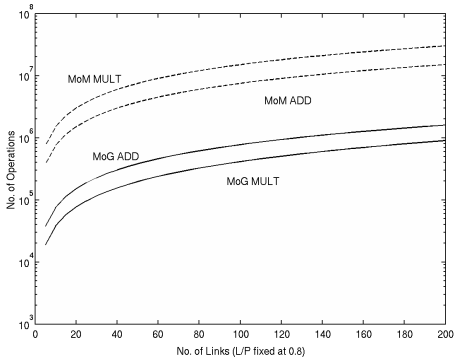


Fig. 5. Number of Operations against Number of Links in network for a fixed value of L/P

Finally, we consider the complexity of inverting H as mentioned in Section 2.1. In [10] it is estimated that inverting a matrix of size $N * N$ takes a number of operations of order N^3. In our scenario, P and L are similar in size so we estimate the operation will be of similar complexity, around $O(L^3)$. As this is a wired scenario, we assume the inversion takes place once as the topology remains fixed throughout; however, we note that were it a wireless scenario with mobile nodes then this would be a more significant contribution to overall complexity.

4. CONCLUSION & FUTURE WORK

In this paper we have presented a comparison of two methods of delay estimation and bottleneck-link discovery for use in wired network tomography. We have shown that the parametric estimation (MoG) method provides performance comparable with the CDF estimation (MoM) method for a short observation window with a reasonable probe-packet rate. We have seen that MoG is less reliable than MoM for low probe-packet rates. In both cases, MoG has the advantage of a reduced computational complexity. We have also shown that performance in both methods can be improved by increasing the length of the observation window.

In future we will consider more methods, of both parameter estimation and CDF estimation types. We imagine that with an accurate model, a parametric method would be the most reliable even with a low probe-packet rate. For a higher probe-packet rate, greater accuracy can be obtained with a CDF estimation method, however, this is at the cost of increased computational complexity.

5. REFERENCES

[1] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *J. Amer. Stat. Assoc*, vol. 91, no. 433, pp. 365–377, 1996.

[2] A. Coates, A. O. H. III, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Mag.*, vol. 19, no. 3, pp. 47–65, May 2002.

[3] M.-F. Shih and A. Hero, "Unicast inference of network link delay distributions from edge measurements," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, vol. 6, Salt Lake City, UT, May 2001, pp. 3421–3424.

[4] Y. Xia and D. Tse, "Inference of Link Delay in Communication Networks," *IEEE J. Select. Areas Commun.*, vol. 24, no. 12, pp. 2235–2248, Dec. 2006.

[5] R. Caceres, N. G. Duffield, J. Horowitz, and D. F. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Trans. Inform. Theory*, vol. 45, no. 7, pp. 2462–2480, Nov. 1999.

[6] M. J. Coates and R. D. Nowak, "Network tomography for internal delay estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, vol. 6, Salt Lake City, UT, May 2001, pp. 3409–3412.

[7] N. G. Duffield, F. L. Presti, V. Paxson, and D. Towsley, "Inferring link loss using striped unicast probes," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, Anchorage, AK, Apr. 2001, pp. 915–923.

[8] Multicast-based inference of network-internal characteristics (MINC). [Online]. Available: http://gaia.cs.umass.edu/minc/

[9] UCL/VINT/LBNL. network simulator ns (version 2). [Online]. Available: http://www.isi.edu/nsnam/ns/

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, England: Cambridge University Press, 1992, ch. 2.

A Comparison of Delay Estimation & Bottleneck-link Detection Methods for Network Tomography

By Nick Johnson †, John Thompson †, Steve McLaughlin †, Francisco J. Garcia ‡

†Institute for Digital Communications, The University of Edinburgh, Edinburgh, UK
{Nick.Johnson,John.Thompson,Steve.McLaughlin}@ed.ac.uk
‡Agilent Laboratories, Scotland, South Queensferry, Edinburgh, UK
Frankie_garcia@agilent.com

Abstract

Network tomography offers a useful method to identify internal problems in a network using data which can be obtained at the edge. Provided the topology of the network is known then it is possible to recover from the edge measurements some properties of internal links of the network which may not be accessible for any number of reasons - cost, ownership, physical location etc. In this paper we compare estimator and detector methods used to find the bottleneck link in a wired network, that is, the link experiencing the highest delay.

1. Introduction

Computer networks are growing in both size and complexity and with the evolution of the internet can be connected in a near infinite number of ways. Typically, the user is situated at the edge of such a network with the resources they wish to access at another edge; thus, the route taken by data travelling between user and resource can be long and complex. Data packets will interact with other traffic on some or all of the route which will have an affect on their statistical properties such as latency (delay), loss rate and interarrival time. It is desireable to measure the network to detect anomalous values of these statistics; network tomography, introduced in [Coates et al] is one method of doing this. In this work, we concentrate on latency-based tomography, in particular finding the network link with highest latency. This involves estimating the route-level delay distribution from measurements available at the route or path level, converting this to an estimate of the link-level distribution and then detecting which link is most probably the one with highest delay (which we refer to as the bottleneck-link).

It is normal to describe the tomography process in terms of Y, the path-level delay measurements, X, the link-level delay data we cannot measure and H, the routing matrix which describes the relationship between X & Y. It is also normal to number the paths from 1 to P and the links from 1 to L. With this nomenclature, a network is of the form:

$$Y = H \times X \tag{1.1}$$

That is, a network is a connected set of L links which form P routes or paths. We wish to find X so we invert this process thus:

$$X = H^{-1} \times Y \tag{1.2}$$

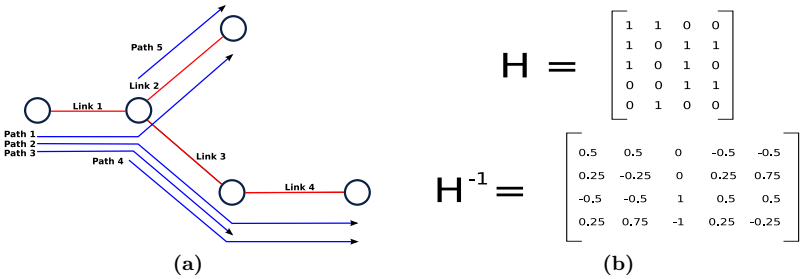


Figure 1: (a) - Network Topology of the small network, originally from [Coates et al] & **(b)** - the H matrix and its pseudo-inverse

Where H^{-1} is the pseudo-inverse of H and conditioned on H being of full rank. This is the least squares (LS) approach to finding X; we express X as a weighted sum of the contributions from each Y with the weights coming from H^{-1}. h_{ij} is the weight assigned to the contribution to link j from path i. This is easier to see in an example so consider Fig 1 where we have $P = 5$ and $L = 4$. Fig 1a shows the topology of the network whilst Fig 1b shows the routing matrix (H) and it's pseudo-inverse (H^{-1}).

The remainder of this paper is organised as follows: in Section 2 we introduce the estimation schemes we are going to compare whilst in Section 3 we introduce detection methods to accompany them. In Section 4 we introduce the parameters used for the test and show some selected results before finally, in Section 5 presenting some conclusions.

2. Estimation Methods

In this section we introduce the estimation methods to be compared in Section 4.

Gaussian Approximation

The use of the Gaussian distribution to model the link delay was suggested in [Shih & Hero] although the authors believed it to have problems with identifiability. Once the mean and variance have been estimated from the N path-level data, they are transformed using LS with weights coming from $|h_{ij}|^2$. Equations 2.1 & 2.2 show the estimation of the mean and variance for path i respectively while Equation 2.3 shows the estimated distribution for link j.

$$\widehat{\mu_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} Y_{ik} \tag{2.1}$$

$$\widehat{\sigma^2_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} (Y_{ik} - \widehat{\mu_{Y_i}})^2 \tag{2.2}$$

We treat the variance as a noise process (multiplying by $|h_{ij}|^2$) in order to stop it becoming negative.

$$X_j = \mathcal{N}(\sum_{i=1}^{P} \widehat{\mu_{Y_i}} \times h_{ij}, \sum_{i=1}^{P} \widehat{\sigma^2_{Y_i}} \times |h_{ij}|^2) = \mathcal{N}(\widehat{\mu_{X_j}}, \widehat{\sigma^2_{X_j}}) \tag{2.3}$$

Sum of Gaussians Type A (SOGA)

To extend the GA method we investigate the idea that the data can be modelled using a Gaussian Mixture Model (GMM). The flexibility gained by using a complex model allows the distribution to be better fitted but at the cost of computation time. We use the well known Expectation-Maximization (EM) algorithm to find the best estimate of the path-level delay distribution as a mixture of J Gaussians, given an initial estimate. With an estimate for each path, we find the combinations of mixtures with the highest joint probability and use this as a metric to select the most likely combinations - here we select the top 1000. As an example, one combination might be, Path 1, 2nd Gaussian, 0.74; Path 2, 1st Gaussian, 0.34; Path 3, 3rd Gaussian, 0.57; giving a metric for this combination of $0.74 \times 0.34 \times 0.34 = 0.143$. We apply LS as in GA to compute the GMM (comprised of mean, variance and weight for 3 Gaussians) for each link. The problem with this method is that we have to compute the joint probabilities for all combinations of GMM. With a small network this is a relatively small number of computations but as the network scales, this value increases rapidly. The number of computations required is $(J^P) * P$ giving 1215 for the small network which increases to 6377292 computations for 12 paths.

Raw Transform (RT)

This is the simplest practical method and is included here as a control method against which to rate the others. The mean value of data over one second (for each path) is taken and LS applied to each block to get an estimate of the link-level data. A histogram is taken of these blocks to get an empirical measure of the distribution function.

Sum of Gaussians Type B (SOGB)

An extension of the RT method, the link-level data is modelled as a GMM once it has been transformed via LS. The advantage is that we do not have the large number of comparisons to perform as in SOGA and we have an analytic expression for the distribution function at the link-level.

Method Of Moments (MOM)

In [Shih & Hero] the authors estimate the Cumulant Generating Function (CGF) of the distribution of delay on each path from individual measurements with a method-of-moments (MOM) estimator. These are passed through the LS algorithm to give a CGF of the delay distributions for each link in the network.
We first construct an estimate of the CGF of path i using N measured delays denoted $Y_{ik}, k = 1...N$,

$$\widehat{M_{Y_i}(t)} = \frac{1}{N} \sum_{k=1}^{N} e^{tY_{ik}} \tag{2.4}$$

Then we use LS to obtain a link-level estimate of the CGF.

$$\widehat{K_{X_j}} = \sum_{i=1}^{P} h_{ij} \times \log(\widehat{M_{Y_i}}) \tag{2.5}$$

3. Detection Methods

To detect the bottleneck-link we must use a detection method which is compatible with the output from the estimation stage. Both methods used in this work require the a-priori selection of a variable δ which itself is an educated guess at the value of delay. The choice of this value is empirical and the accuracy of detection is often highly dependant on it; both major limitations.

CDFmax

To detect the bottlneck-link, we evaluate the link CDFs at a fixed value of δ and pick the link which has lowest value. This is dependant on a good estimate of the CDF to achieve reliable detection but is suitable for any of the parametric methods or empirical methods (ie GA, SOGA, SOGB & RT) used here.

Chernoff Bound

To detect the bottleneck, we impose a Chernoff upper bound on the link CGFs and it is therefore the most suitable choice to use with MOM. The bottleneck is the link with the highest probability (P_j) of exceeding the delay threshold (δ).
In [Shih & Hero] this is expressed as:

$$P_j = P(X_j \geq \delta) \leq e^{-t\delta} E[e^{tX_j}] \tag{3.1}$$

4. Selected Simulation Results

Simulation Setup

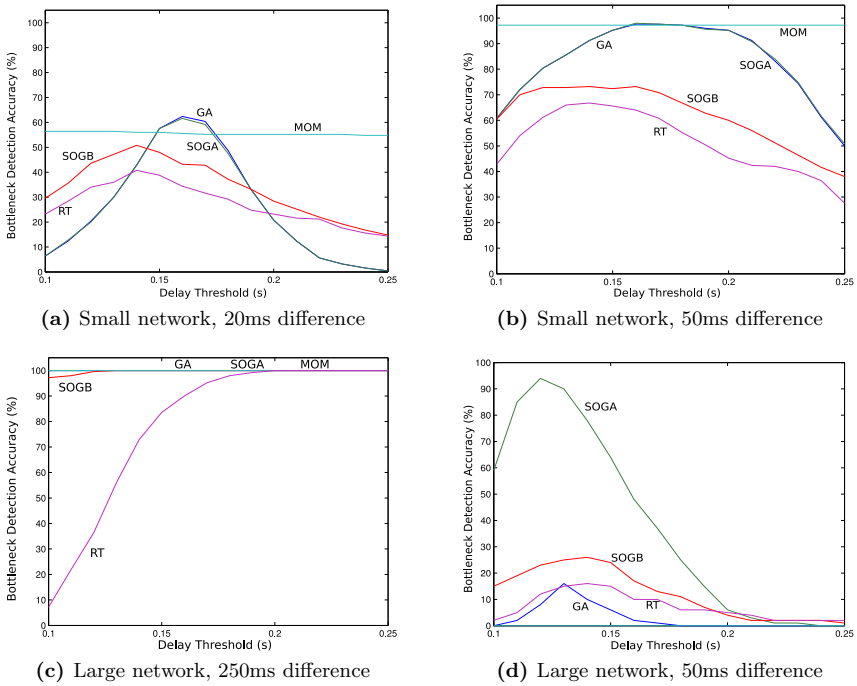
To test the methods we used an ns2 [ns2] simulation to model a wired network with unicast probe-path traffic. The topology of the small network is as shown in Fig 1a and the key parameters show in Table 1. On each link we add a delay to each packet to force a situation where one link has a higher latency than the others for the methods to detect. Aside from the added delay, other delays encountered by packets come from self-congestion due to background traffic in the form of queueing and processing time at each node.
We tested two sizes of network to observe robustness with regards to scaling of the network and used different values of added bottleneck delay to test responsiveness under conditions where the added delay was similar to that of the background delay. Background traffic on each link is formed by combining a number of exponentially distributed UDP and a number of TCP traffic sources in a similar manner to that used in [Coates et al].

Discussion of Results

Figure 2 shows the detection accuracy plotted against the values of δ between 0.1 and 0.25 seconds; they therefore show the sensitivity of each of the methods to the choice of δ when the bottleneck delay is fixed. Where the difference between the bottleneck and the next worst link is small (2a) the peak accuracy is quite sensitive to choice of δ for all methods except MOM. As the difference increases, so does the peak accuracy range (in terms of δ) and value (2b); this is not unexpected as it is *easier* to select the bottleneck from the other links. With a larger network, a larger difference is required before the same detection accuracy can be obtained (compare 2b & 2d). Here GA exhibits quite poor performance despite performing well in the small network whilst MOM fails to select the bottleneck at all. If the difference increases again (2c) then we see MOM, GA, SOGA and SOGB performing well regardless of choice of δ; only RT shows some

Parameter	**Small Simulation**	**Large Simulation**
Added Delay on bottleneck link	20ms, 50ms	50ms, 250ms
Added Delay on normal links	10ms, 80ms, 100ms	100ms
Bandwidth on each link	1 Mb	1Mb
Simulation Time	5000s	5000s
Number of Paths, P	5	12
Number of Link, L	4	9
CGF Parameter, t	20	20
Number of probe packets, N	25000	25000
Probe packet rate	2 Kb/s	2Kb/s
Probe packet size	40 Bytes	40 Bytes

Table 1: Key Simulation Parameters



(a) Small network, 20ms difference

(b) Small network, 50ms difference

(c) Large network, 250ms difference

(d) Large network, 50ms difference

Figure 2: Detection Accuracy against choice of delay threshold (δ) for four scenarios

sensitivity. Overall, we see GA and its more complex version, SOGA, perform well in each of the scenarios. SOGA exhibits good performance in the larger network with small delay difference which is the most challenging of the four scenarios. SOGB outperforms our control method (RT) which may indicate that a good model of the tail of the distribution can have a positive effect; being essentially a histogram, RT has poor performance here. We also note that MOM does not depend on choice of δ and therefore has constant performance in each of the scenarios; this is not to suggest that it is a better choice as a good selection of δ can provide SOGA or GA with better performance, especially when

the difference is low. Despite its poor performance in 2d GA remains a robust choice of method as it has much lower computational load compared to SOGA which may be more important than a small increase in accuracy in an application where the bottleneck is likely to be much worse than the next-worst link.

5. Conclusion

We have provided a comparison of five methods for estimating the bottleneck-link using delay-based network tomography. We have seen that whilst there are advantages to using a non-parametric method, namely its ability to model an arbitrary distribution, a reasonable choice of close fitting paramteric distribution can produce more accurate overall results.

REFERENCES

Coates, A., Hero, A. O., Nowak, R., Yu, B. 2002 Internet Tomography *IEEE Sig. Proc. Mag.* vol. 19, no. 3, pp. 47–65.

Shih, M-F., Hero, A. O. 2001 Unicast inference of network link delay distributions from edge measurements *Proc. ICASSP 2001* vol. 6, pp. 3421–3424

UCL/VINT/LBNL network simulator ns2, Available online http://www.isi.edu/nsnam/ns/

A COMPARISON OF SOME BOTTLENECK-LINK DETECTION METHODS FOR NETWORK TOMOGRAPHY

Nick Johnson †, John Thompson †, Steve McLaughlin †and Francisco J. Garcia ‡

†
Institute for Digital Communications
Joint Research Institute for Signal & Image Processing
School of Engineering
The University of Edinburgh
Edinburgh, EH9 3JL, UK
{Firstname . Surname}@ed.ac.uk

‡
Agilent Laboratories, (Scotland)
South Queensferry
Edinburgh, UK

Frankie_ garcia@agilent.com

ABSTRACT

Network tomography offers a useful method to identify internal problems in a network using data which can be obtained at the network's edge. Provided the topology of the network is known then it is possible to recover from the edge measurements some properties of internal links of the network which may not be accessible for any number of reasons - cost, ownership, physical location etc. In this paper we introduce two new estimation algorithms based on the Pearson type-1 distribution and compare these with existing estimator and detector algorithms used to find the bottleneck link in a wired network, that is, the link experiencing the highest delay.

1. INTRODUCTION

Computer networks are growing in both size and complexity and with the evolution of the internet can be connected in a near infinite number of ways. Typically, the user is situated at the edge of such a network with the resources they wish to access at another edge; thus, the route taken by data travelling between user and resource can be long and complex. Data packets will interact with other traffic on some or all of the route which will have an affect on their statistical properties such as latency (delay), loss rate and interarrival time. It is desireable to measure the network to detect anomalous values of these statistics; network tomography, introduced in [1] is one method of doing this.

In this work, we concentrate on latency-based tomography, in particular finding the network link with highest latency. We define a network link as a connection between two nodes or routers in a network and a route or path as a connected set of links, this is illustrated in Figure 1. We estimate the route-level delay distribution from measurements available at the route or path level, convert this to an estimate of the link-level distribution and then detect which link is most probably the one with highest delay (which we refer to as the bottleneck-link). There are many approaches to this problem, some of which use parametric distributions for link and path estimates such as exponential and exponential mixture distributions [2] or Gaussian mixture distributions [3]. Alternatives use EM based approaches [4] whilst some are based upon particle filters [5]. Some of the most recent and potentially most efficient, in terms of volume of data required to detect any bottleneck, are based upon compressed sensing [6]. Our contribution is in extending two existing estimation algorithms using the Pearson type-1 distribution to form two new estimation algorithms which may offer a computational saving over the originals.

The remainder of this paper is organised as follows: in Section 2 we introduce our system model, in Section 3 we introduce the estimation algorithms we are going to compare and in Section 4 we introduce detection algorithms to accompany them. In Section 5 we introduce the parameters used for the test and show some selected results before finally, in Section 6 presenting some conclusions.

2. SYSTEM MODEL

It is possible to describe tomography using some basic algebra; here, we attempt to remain consistent with other works [1], [7] in our nomenclature. We refer to the path-level delay measurements as Y, the link-level delay data we cannot measure but wish to estimate as X and the routing matrix (which describes the relationship between X and Y) as H. The paths in any network are numbered from 1 to P while the links are numbered from 1 to L. Using this notation, we can think of a network as:

$$Y = H \times X \tag{1}$$

This implies that the P routes (or paths, the terms are often used interchangeably) in a network are simply an interconnected set of L links: H, the routing matrix, describes how the links are connected to form the paths using a 1 to indicate the the link is part of the route and a 0 to indicate otherwise. Since our desire is not to gather information on Y (which we can measure directly) but on X, we invert our equation:

$$X = H^{-1} \times Y \tag{2}$$

We can see that this is the well known least squares (LS) method applied to finding X where H^{-1} is the L2 pesudo-inverse of H. We seek to find an estimate of X as a weighted sum of the contributions from each Y with the weights coming from H^{-1}. We define h_{ij} as the weight assigned to the contribution to link j from path i.

It is perhaps easiest to illustrate this using a small example so consider a networl which has a topology such as that shown in Figure 1 with $P = 5$ and $L = 4$. Link 1 and link2 form path 1 so the routing matrix has a 1 in the first two columns of

the first row: the columns of the routing matrix correspond to the links while the rows correspond to the paths. Equation 3 shows the routing matrix (H) while equation 4 shows it's pseudo-inverse (H^{-1}).

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (3)$$

$$H^{-1} = \begin{bmatrix} 0.50 & 0.50 & 0 & -0.50 & -0.50 \\ 0.25 & -0.25 & 0 & 0.25 & 0.75 \\ -0.50 & -0.50 & 1 & 0.50 & 0.50 \\ 0.25 & 0.75 & 1 & 0.25 & -0.25 \end{bmatrix} \quad (4)$$

The tomography problem presented above can be reduced in it's most basic format to this: trying to estimate the link delay CDF (X) using measurements taken at the route level (Y) and knowledge of how the network is constructed (H). This can be used to determine the link with the highest delay which can be useful for monitoring and control applications.

3. ESTIMATION ALGORITHMS

In this section we introduce four algorithms to estimating the statistical properties of the links based on the data acquired at the path level; the results are used with a complementary detection algorithm described in Section 4 and the performance shown in Section 5.

3.1 Gaussian Approximation

The use of a single Gaussian distribution to model link delay was suggested in [2] although the authors believed it to have problems with identifiably. We also treat the path delay as having a Gaussian distribution so that the mean and variance are estimated from N path data and subsequently transformed to provide an estimate of the mean and variance of the link using LS with weights coming from $|h_{ij}|^2$. Equations (5) & (6) show the estimation of the mean and variance for path i respectively while equation (7) shows the estimated distribution for link j.

$$\widehat{\mu_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} Y_{ik} \quad (5)$$

$$\widehat{\sigma^2_{Y_i}} = \frac{1}{N}\sum_{k=1}^{N} (Y_{ik} - \widehat{\mu_{Y_i}})^2 \quad (6)$$

We treat the variance as a noise process and so square the weights ($|h_{ij}|^2$) to preserve the positivity of the link estimate.

$$X_j = \mathcal{N}(\sum_{i=1}^{P} \widehat{\mu_{Y_i}} \times h_{ij}, \sum_{i=1}^{P} \widehat{\sigma^2_{Y_i}} \times |h_{ij}|^2) = \mathcal{N}(\widehat{\mu_{X_j}}, \widehat{\sigma^2_{X_j}}) \quad (7)$$

Algorithm 1 GA estimation algorithm

1: **for** $i = 1$ to P **do**
2: fit single Gaussian to path, i, using equations (5) & (6)
3: **end for**
4: **for** $j = 1$ to L **do**
5: estimate PDF of link j using path estimates 1 to P using LS as per equation (7)
6: **end for**

3.2 Method Of Moments (MOM)

In [1] and [7] the authors estimate the Cumulant Generating Function (CGF) of the distribution of delay on each path from individual measurements with a method-of-moments (MOM) estimator. These are passed through the LS algorithm to give a CGF of the delay distributions for each link in the network.
We first construct an estimate of the CGF of path i using N measured delays denoted $Y_{ik}, k = 1...N$,

$$\widehat{M_{Y_i}(t)} = \frac{1}{N}\sum_{k=1}^{N} e^{tY_{ik}} \quad (8)$$

Then we use LS to obtain a link-level estimate of the CGF.

$$\widehat{K_{X_j}} = \sum_{i=1}^{P} h_{ij} \times \log(\widehat{M_{Y_i}}) \quad (9)$$

Algorithm 2 MOM estimation algorithm

1: **for** $i = 1$ to L **do**
2: estimate CGF of path, i, using equation (8)
3: **end for**
4: **for** $j = 1$ to L **do**
5: estimate CGF of link j using path estimates 1 to P using LS as per equation (9)
6: **end for**

3.3 Pearson type-1 Distribution (PRS)

With GA we use the first two moments to model the data but we suspect that this might not allow enough flexibility to give an accurate model so we use a Pearson type-1 distribution to make use of the first four moments. The algorithm is similar to GA but with the addition of skewness and kurtosis which we recall in equations (10) and (11) respectively.

$$\widehat{\mu_3} = \frac{1}{N}\sum_{k=1}^{N} E((Y_{ik} - \mu_{i,1})^3)/\sigma^3 \quad (10)$$

$$\widehat{\mu_4} = \frac{1}{N}\sum_{k=1}^{N} E((Y_{ik} - \mu_{i,1})^4)/\sigma^4 - 3 \quad (11)$$

LS is applied to the four estimates for each path as in GA and similarly results in a set of estimates for a Pearson distribution for each link. We recall that the Pearson has a condition in that $\mu_4 \geq \mu_3^2 + 1$ which we find may not always occur due to the transformation in LS; in this situation we modify the values of μ_3 so that the condition is satisfied.

3.4 Pearson - Method Of Moments (P-MOM)

The problem with MOM is that it does not produce an estimate of the CDF (or PDF) of the delay for each link which

Algorithm 3 PRS estimation algorithm

1: **for** $i = 1$ to P **do**
2: fit a Pearson type-1 distribution to path i, using equations (5), (6), 10 & 11.
3: **end for**
4: **for** $j = 1$ to L **do**
5: estimate CDF of link j using path estimates 1 to P using LS in a similar manner to equation (7)
6: **end for**

is preferable to the CGF as it may give more insight into the operation of the network. To overcome this, we fit a Pearson type-1 distribution to the first four parameters estimated by MOM.

Algorithm 4 P-MOM estimation algorithm

1: **for** $i = 1$ to L **do**
2: estimate CGF of path, i, using equation 8
3: **end for**
4: **for** $j = 1$ to L **do**
5: estimate CGF of link j using path estimates 1 to P using LS as per equation 9
6: **end for**
7: **for** $j = 1$ to L **do**
8: generate Pearson type-1 distribution using cumulants from CGF for each link j
9: **end for**

4. DETECTION ALGORITHMS

To detect the bottleneck-link we employ a detection algorithm compatible with the estimator output. Both detection algorithms used here require the a-priori selection of a variable, δ, which is an educated guess at the value of delay. The choice of δ is empirical and detection accuracy is often dependant on it; both major limitations.

4.1 CDFmax

We evaluate the link CDFs at a fixed value of δ and call this P_j and pick as bottleneck the link with lowest P_j. This relies on a good CDF estimate to achieve reliable detection but is suitable for any of the parametric algorithms - ie GA, PRS and P-MOM.

$$P_j = \arg\min_j \left(cdf_j(\delta) \right); \quad j\varepsilon\{1,2,\ldots,L\} \qquad (12)$$

4.2 Chernoff Bound

We impose a Chernoff upper-bound on the link CGFs and select as bottleneck the link with the highest probability (P_j) of exceeding the delay threshold (δ).
In [1] and [7] this is expressed as:

$$P_j = P(X_j \geq \delta) \leq e^{-t\delta} E[e^{tX_j}] \qquad (13)$$

5. SELECTED SIMULATION RESULTS

5.1 Simulation Setup

To test all the methods (where a method is a combination of estimation and detection algorithms, named after the estimator) we use an ns2 [8] simulation to model a wired network with unicast probe-path traffic. The topology of the 5-node network is shown in Figure 1 while the 10 node network has a larger but similar structure: the key parameters

Parameter	Value
Bottleneck delay (5 node)	120ms, 150ms
Bottleneck delay (10node)	150ms, 200ms, 250ms
Normal link delay (5 node)	10ms, 80ms, 100ms
Normal link delay (10 node)	100ms
Link bandwidth	1 Mb
Simulation time	5000s
Number of paths, P (5 node)	5
Number of paths, P (10 node)	12
Number of links, L (5 node)	4
Number of links, L (10 node)	9
CGF parameter, t	20
Number of probe packets, N	25000
Probe packet rate	2 Kb/s
Probe packet size	40 Bytes

Table 1: Key Simulation Parameters

are shown in Table 1. Background traffic on each link is formed by combining exponentially-distributed constant-bit-rate UDP and TCP traffic sources similar to those in [1]. The rates of the UDP sources are chosen to ensure each link is between 70% and 80% utilized; the TCP sources adjust their rate to achieve maximum throughput ensuring links operate at peak capacity. This is important because we want to ensure that packets on the network experience some delay due to congestion. The delay is manifested as queueing and processing time at each node. In addition, on each link we add a delay to each packet so that one link has a delay higher than the others for our methods to detect. This ensures we have a scenario where we know the bottleneck and can control the degree of detection difficulty.

5.2 Discussion of Results

We compare the performance of the estimator and detector combinations in terms of detection accuracy and computational complexity in six different scenarios:

- A 5 node topology with 20ms separation (Figure 2a)
- A 5 node topology with 50ms separation (Figure 2b)
- A 10 node topology with 50ms separation (Figure 3a)
- A 10 node topology with 150ms separation (Figure 3b)
- A 10 node topology with 250ms separation (Figure 3c)

In Figure 2a PRS outperforms GA in terms of peak accuracy by about 20%, however both methods show sensitivity to choice of δ; this may indicate information loss when using two moments in GA. MOM is less sensitive to δ than P-MOM although both offer a similar level of accuracy of between 50 and 60% which we assign to the MOM-based parameter estimation.

In Figure 2b the separation has increased: GA now outperforms PRS, the peak accuracy is greater by 6% while the range of δ over which it has high accuracy has increasedr. MOM and P-MOM show similar performance with the difference being slightly less than in Figure 2a.

If the separation increases to a larger value, ie 150ms, then all methods converge to 100% accuracy. This is not unexpected as the separation is high (greater than twice the delay of the 2nd worst link) and it should be easy to detect a bottleneck even if the estimation is poor.

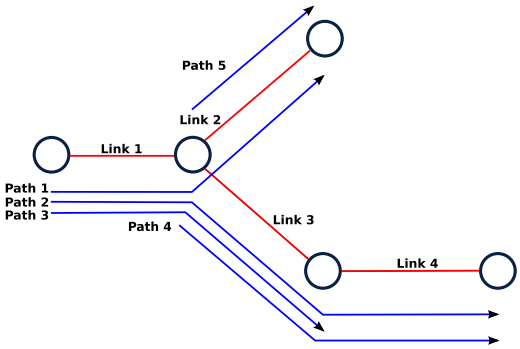
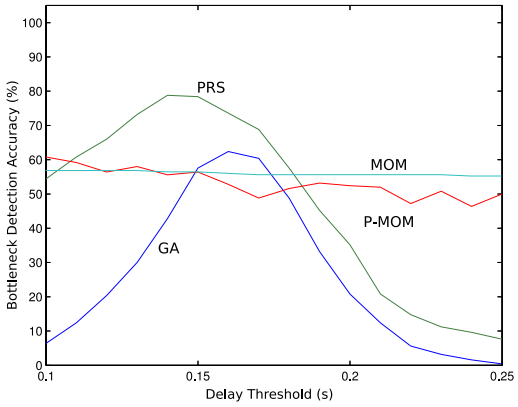
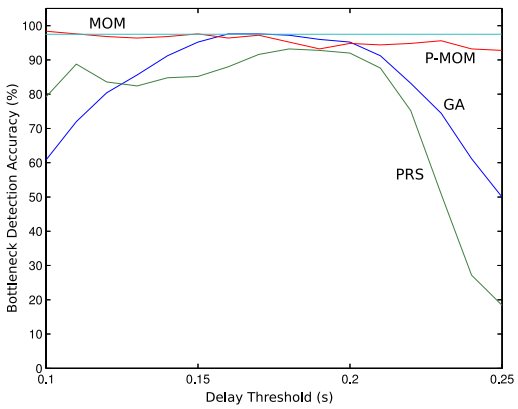


Figure 1: Network topology showing paths and links, originally from [1]

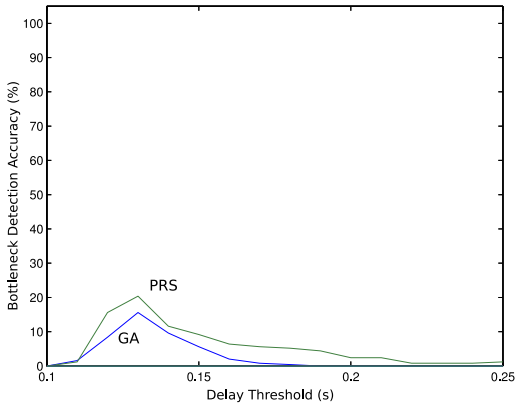


(a) 20ms separation

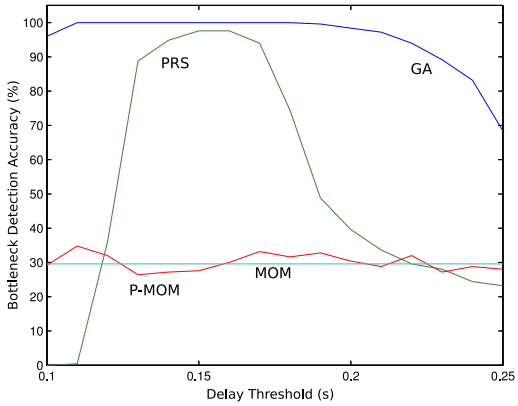


(b) 50ms separation

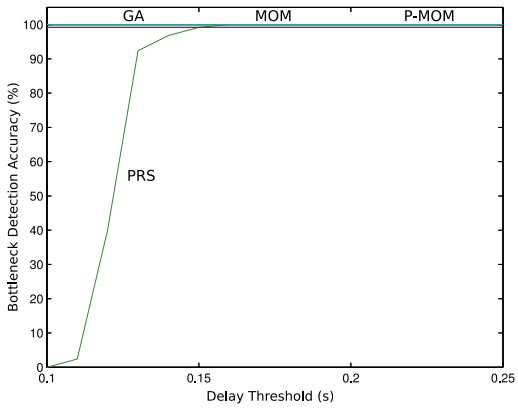
Figure 2: Detection Accuracy against choice of delay threshold (δ) for two 5 node scenarios



(a) 50ms separation



(b) 100ms separation



(c) 250ms separation

Figure 3: Detection Accuracy against choice of delay threshold (δ) for three 10 node scenarios

Algorithm	MULT
GA	$L(87+3P)+P(N+2)$
PRS	$P(11N+5L+2)+165L$
MOM	$Pt(2N+1)+L(5t+2)$
P-MOM	$167L+4P(N-1)$

Algorithm	ADD
GA	$30L+L(P-1)+3PN$
PRS	$8PN+4L(P-1)+52L+(L-1)!$
MOM	$PtN+LA+(L-1)!$
P-MOM	$L(P+52)+4PN+(L-1)!$

Table 2: Formulae for number of MULT and ADD operations required for estimation/detection of 20s of data in the 5 node network.

Algorithm	MULT	ADD
GA	5418	15136
PRS	55770	40278
MOM	200508	116006
P-MOM	20648	20234

Table 3: Numerical results for the formulae in Table 2 using values from Table 1 for the scenario in Figure 2a with $N = 1000$.

Figure 3a shows the performance of the methods in the 10 node scenario. Both MOM and P-MOM are unable to correctly detect the bottlneck while GA and PRS exhibit poor performance although PRS is the more accurate. As the topology scales, the separation required for a fixed level of accuracy increases.

In Figure 3b the accuracy of GA is comparable to Figure 2b: the range of PRS is narrower and peak accuracy is slightly less than GA. MOM and P-MOM exhibit similar accuracy of around 30% with P-MOM having some variance as in the 5 node scenarios.

In Figure 3c the accuracy of GA, MOM and P-MOM has increased to the maximum which is expected given the large separation. Interestingly, the PRS response has a lower tail similar to that in Figure 3b showing sensitivity to δ.

Table 2 shows the number of multiplication (MULT) and add (ADD) operations required to perform one estimation and detection on a block of data (around 20s of data) using the scenario in Figure 2a. GA is the most computationally efficient as it has a low number of parameters and scales with the number of samples, N. PRS scales with N but has a more complex CDF and a greater number of parameters than GA. MOM scales with N and t which increases the complexity for even a small value of t. P-MOM requires t equal to 4 so offers a complexity saving compared to MOM. Table 3 expresses this numerically with the parameters as in Table 1 but with $N = 1000$. This illustrates that MOM is the least computationally efficient followed by PRS. Crucially we see that P-MOM combines the efficient estimation of MOM with the efficient output evaluation of PRS requiring one fifth the number of ADDs and one tenth the number of MULTs of MOM.

With large separation, all methods are able to correctly identify the bottleneck in a network. As separation decreases, the parametric methods (GA and PRS) appear more robust, however they are sensitive to choice of δ making them unsuitable in an environment where little is known about the normal operating conditions of the network. A non-parametric method such as MOM has reduced sensitivity to δ but its output may not be compatible with all types of detection algorithm. A hybrid approach, P-MOM, which uses the Pearson distribution as output sacrifices some of the accuracy of MOM but gains reduced sensitivity and decreased computational cost.

6. CONCLUSION

In this paper we compared four network tomography methods, two of which make use of the Pearson distribution and are introduced here, to perform bottleneck-link detection. We saw that a non-parametric algorithm (MOM) provides consistent, reliable performance which is not constrained by an *a-priori* choice of delay threshold. However it was not as robust as a parametric algorithm (GA and PRS) in low separation scenarios and was computationally expensive. Robustness has to be carefully traded with computational cost when considering overall suitability, especially in a real-time environment and we conclude that it may be preferential to sacrifice robustness for a reduction in compute time (GA). Where sensitivity to δ is a concern and a CDF output is desireable then our hybrid method, P-MOM, offers a solution which combines the flexibility of PRS with the consistent performance of MOM.

REFERENCES

[1] A. Coates, A. O. Hero III, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, 2002.

[2] Y. Xia and D. Tse, "Inference of link delay in communication networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2235–2248, 2006.

[3] N. Johnson, J. Thompson, S. McLaughlin, and F. Garcia, "A comparison of delay estimation & bottleneck-link detection methods for network tomography," in *Proceedings of the 8th IMA conference on Mathematics in Signal Processing*, Cirencester, UK, Dec 2008.

[4] Y. Sun, D. Li, and H. Sun, "Network Tomography and Improved Methods for Delay Distribution Inference," in *The 9th International Conference on Advanced Communication Technology*, vol. 2, Gangwon-Do, Feb. 2007, pp. 1433–1437.

[5] M. J. Coates and R. D. Nowak, "Sequential monte carlo inference of internal delays in nonstationary data networks," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 366–376, 2002.

[6] M. Coates, Y. Pointurier, and M. Rabbat, "Compressed Network Monitoring," in *SSP '07. IEEE/SP 14th Workshop on Statistical Signal Processing*, Madison, WI, USA, Aug. 2007, pp. 418–422.

[7] M.-F. Shih and A. Hero, "Unicast inference of network link delay distributions from edge measurements," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 6, 2001, pp. 3421–3424.

[8] UCL/VINT/LBNL. network simulator ns (version 2). [Online]. Available: http://www.isi.edu/nsnam/ns/

References

[1] A. Coates, A. O. Hero III, R. Nowak, and B. Yu, "Internet Tomography," *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, 2002.

[2] J. Kruithof, "Telefoonverkeersrekening," *De Ingenieur*, vol. 52, no. 8, pp. E15 – E25, 1937.

[3] M.-F. Shih and A. Hero, "Unicast Inference of Network Link Delay Distributions from Edge Measurements," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 6, 2001, pp. 3421–3424.

[4] M. Coates, Y. Pointurier, and M. Rabbat, "Compressed Network Monitoring," in *SSP '07. IEEE/SP 14th Workshop on Statistical Signal Processing*, Madison, WI, USA, Aug. 2007, pp. 418–422.

[5] Y. Xia and D. Tse, "Inference of Link Delay in Communication Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2235–2248, 2006.

[6] Y. Sun, D. Li, and H. Sun, "Network Tomography and Improved Methods for Delay Distribution Inference," in *The 9th International Conference on Advanced Communication Technology*, vol. 2, Gangwon-Do, Feb. 2007, pp. 1433–1437.

[7] G. Danezis, *The Traffic Analysis of Continuous-time Mixes*, ser. Lecture Notes in Computer Science. Springer, 2004, vol. 3424, pp. 35–50.

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of 13th USENIX Security Symposium*, August 2004.

[9] ——. TOR: The Onion Router. Tor Project/EFF. [Online]. Available: http://www.torproject.org

[10] International Telecommunication Union. (1994, July) Open Systems Interconnection - Basic Reference Model: The Basic Model. ITU-T. [Online]. Available: http://www.itu.int/rec/T-REC-X.200-199407-I/en

[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 1991.

[12] R. A. Leibler and S. Kullback, "On Information and Sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 19–86, 1951.

[13] International Telecommunication Union. (1993, March) Integrated Services Digital Networks (ISDN). ITU-T. [Online]. Available: http://www.itu.int/rec/T-REC-I.120-199303-I/en

[14] N. Duc, "ISDN Terminals and Integrated Services Delivery," *IEEE Journal on Selected Areas in Communications*, vol. 4, no. 8, pp. 1188–1192, Nov 1986.

[15] ETSI. (2008, December) High Speed Circuit Switched Data (HSCSD). 3GPP. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23034.htm

[16] L. Roberts, "The Evolution of Packet Switching," *Proceedings of the IEEE*, vol. 66, no. 11, pp. 1307–1313, Nov. 1978.

[17] J. Postel. (1980, January) Internet Protocol. University of Southern California / Information Sciences Institute. [Online]. Available: http://tools.ietf.org/rfc/rfc760.txt

[18] ——. (1981, September) Internet Protocol. IETF. [Online]. Available: http://www.rfc-editor.org/rfc/rfc791.txt

[19] V. G. Cerf and R. E. Kahn, "A Protocol for Packet Network Intercommunication," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637–648, May 1974.

[20] J. Postel. (1981, September) Transmission Control Protocol. IETF. [Online]. Available: http://tools.ietf.org/rfc/rfc793.txt

[21] R. Braden. (1989, October) Requirements for Internet Hosts – Communication Layers. IETF. [Online]. Available: ftp://ftp.rfc-editor.org/in-notes/rfc1122.txt

[22] International Telecommunication Union. (1996, October) Interface Between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit. ITU-T. [Online]. Available: http://www.itu.int/rec/T-REC-X.25/en

[23] ——. (1999, February) B-ISDN Asynchronous Transfer Mode Functional Characteristics. ITU-T. [Online]. Available: http://www.itu.int/rec/T-REC-I.150/en

[24] G. Fairhurst and L. Wood. (2002, August) Advice to Link Designers on Link Automatic Repeat reQuest (ARQ). IETF. [Online]. Available: http://www.ietf.org/rfc/rfc3366.txt

[25] M. Allman, V. Paxson, and W. Stevens. (1999, April) TCP Congestion Control. The Internet Society / IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2581.txt

[26] M. Allman, V. Paxson, and E. Blanton. (2009, September) TCP Congestion Control. IETF. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5681.txt

[27] S. Low, L. Peterson, and L. Wang, "Understanding TCP Vegas: Theory and Practice," Princeton University Computer Science Department, Tech. Rep., February 2000. [Online]. Available: http://www.cs.princeton.edu/research/techreps/TR-616-00

[28] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[29] S. Floyd, T. Henderson, and A. Gurtov. (2004, April) The NewReno Modification to TCP's Fast Recovery Algorithm. The Internet Society / IETF. [Online]. Available: http://www.ietf.org/rfc/rfc3782.txt

[30] J. Postel. (1980, August) User Datagram Protocol. [Online]. Available: http://tools.ietf.org/rfc/rfc768.txt

[31] J. Klensin. (2003, February) Role of the Domain Name System (DNS). IETF. [Online]. Available: http://www.ietf.org/rfc/rfc3467.txt

[32] P. Mockapetris. (1987, November) Domain Names - Concepts and Facilities. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc1034.txt

[33] R. Droms. (1997, March) Dynamic Host Configuration Protocol. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2131.txt

[34] CAIDA. (2009, May) Analyzing UDP Usage in Internet Traffic. CAIDA. [Online]. Available: http://www.caida.org/research/traffic-analysis/tcpudpratio/

[35] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. (2008, July) OSPF for IPv6. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc5340.txt

[36] D. Oran. (1990, February) OSI IS-IS Intra-domain Routing Protocol. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc1142.txt

[37] Y. Rekhter, T. Li, and S. Hares. (2006, January) A Border Gateway Protocol 4 (BGP-4). IETF. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt

[38] G. Almes, S. Kalidindi, and M. Zekauskas. (1999, September) A One-Way Delay Metric for IPPM. Network Working Group, IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2679.txt

[39] ——. (1999, September) A One-Way Packet Loss Metric for IPPM. Network Working Group, IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2680.txt

[40] C. U. Castellanos, D. L. Villa, O. M. Teyeb, J. Elling, and J. Wigard, "Comparison of Available Bandwidth Estimation Techniques in Packet-switched Mobile Networks," in *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, Sept. 2006, pp. 1–5.

[41] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 879–894, Aug. 2003.

[42] S. Ekelin, M. Nilsson, E. Hartikainen, A. Johnsson, J.-E. Mangs, B. Melander, and M. Bjorkman, "Real-time Measurement of End-to-end Available Bandwidth using Kalman Filtering," *2006. NOMS 2006. 10th IEEE/IFIP Network Operations and Management Symposium*, pp. 73–84, 2006.

[43] Y. Cheng, V. Ravindran, A. Leon-Garcia, and H.-H. Chen, "New Exploration of Packet-pair Probing for Available Bandwidth Estimation and Traffic Characterization," in *Proc. IEEE International Conference on Communications ICC '07*, 2007, pp. 588–594.

[44] K. Wang, Z.-C. Li, F. Yang, Q. Wu, and J.-P. Bi, "Experiment and Analysis of Active Measurement for Packet Delay Dynamics," in *Lecture Notes in Computer Science*, X. Lu and W. Zhao, Eds. SPRINGER-VERLAG, 2005, no. 3619, pp. 1063–1072.

[45] V. Jacobson. pathchar. [Online]. Available: http://www.caida.org/tools/utilities/others/pathchar

[46] S. Savage. sting. [Online]. Available: http://www.cs.washington.edu/homes/savage/sting/

[47] C. Dovrolis. pathrate. [Online]. Available: http://www.pathrate.org

[48] InMon Corporation. sflow. InMon Corporation. [Online]. Available: http://www.sflow.org

[49] Cooperative Association for Internet Data Analysis. [Online]. Available: http://www.caida.org/home/

[50] Y. Vardi, "Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data," *J. Amer. Stat. Assoc*, vol. 91, no. 433, pp. 365–377, March 1996. [Online]. Available: http://www.jstor.org/stable/2291416

[51] R. S. Krupp, "Properties of Kruithof's Projection Method," in *The Bell System Technical Journal*. American Telephone and Telegraph Company, 1979, vol. 58, no. 2, pp. 517 – 538.

[52] A. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: http://www.jstor.org/stable/2984875

[53] J. A. Blimes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," 1998.

[54] J. Cao, D. David, S. V. Wiel, and B. Yu, "Time-varying Network Tomography: Router Link Data," *Journal of the American Statistical Association*, vol. 95, no. 452, pp. 1063–1075, 2000. [Online]. Available: http://www.jstor.org/stable/2669743

[55] Y. Tsang, M. Coates, and R. Nowak, "Passive Network Tomography using EM Algorithms," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 3, 2001, pp. 1469–1472 vol.3.

[56] N. G. Duffield and F. Lo Presti, "Network Tomography from Measured End-to-end Delay Covariance," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 978–992, 2004.

[57] M. J. Coates and R. D. Nowak, "Sequential Monte Carlo Inference of Internal Delays in Nonstationary Data Networks," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 366–376, 2002.

[58] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. Springer, 2001, iSBN 0-387-95146-6.

[59] K. Bharath-Kumar and J. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Transactions on Communications*, vol. 31, no. 3, pp. 343–351, 1983.

[60] M.-F. Shih and A. O. Hero III, "Unicast-Based Inference of Network Link Delay Distributions with Finite Mixture Models," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2219–2228, 2003.

[61] A. Chen, J. Cao, and T. Bu, "Network Tomography: Identifiability and Fourier Domain Estimation," in *Proc. INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 1875–1883.

[62] R. Penrose, "A Generalized Inverse for Matrices," in *Proceedings of the Cambridge Philosophical Society*, ser. Proceedings of the Cambridge Philosophical Society, vol. 51, July 1955, pp. 406–413.

[63] E. H. Moore, "On the Reciprocal of the General Algebraic Matrix," *Bulletin of the American Mathematical Society*, vol. 26, pp. 394–395, 1920.

[64] G. de Villiers, "An Algorithm for Reconstructing Positive Images from Noisy Data," in *Proceedings of EUSIPCO 96*, 1996.

[65] N. Hohn and D. Veitch, "Inverting Sampled Traffic," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, Feb. 2006.

[66] R. W. Wolff, "Poisson Arrivals See Time Averages," *Operations Research*, vol. 30, no. 2, pp. 223–231, Mar–Apr 1982. [Online]. Available: http://www.jstor.org/stable/170165

[67] F. Baccelli, S. Machiraju, D. Veitch, and J. Bolot, "The Role of PASTA in Network Measurement," in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2006, pp. 231–242.

[68] F. Baccelli, S. Machiraju, D. Veitch, and J. C. Bolot, "On Optimal Probing for Delay and Loss Measurement," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2007, pp. 291–302.

[69] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. (1998, May) Framework for IP Performance Metrics. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2330.txt

[70] C. H. Foh and M. Zukeman, "Poisson Arrivals See Time Averages," online. [Online]. Available: http://web.mysites.ntu.edu.sg/aschfoh/public/Sharedg/handout-pasta.pdf

[71] M. M. B. Tariq, A. Dhamdhere, C. Dovrolis, and M. Ammar, "Poisson Versus Periodic Path Probing," in *IMC '05: Proceedings of the 5th ACM SIGCOMM on Internet measurement*. New York, NY, USA: ACM Press, 2005.

[72] V. Raisanen, G. Grotefeld, and A. Morton. (2002, November) Network Performance Measurement with Periodic Streams. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc3432.txt

[73] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, June 1995.

[74] S. Floyd and V. Jacobson, "The Synchronization of Periodic Routing Messages," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 122–136, April 1994.

[75] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *Networking, IEEE/ACM Transactions on*, vol. 2, no. 1, pp. 1 –15, feb 1994.

[76] M. Crovella and A. Bestavros, "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes," *Networking, IEEE/ACM Transactions on*, vol. 5, no. 6, pp. 835 –846, dec 1997.

[77] Athanasios Papoulis, *Probability, Random Variables, and Stochastic Processes*, 9th ed. McGraw-Hill, 1965.

[78] B. Puig, F. Poirion, and C. Soize, "Non-Gaussian Simulation using Hermite Polynomial Expansion: Convergences and Algorithms," *Probabilistic Engineering Mechanics*, vol. 17, no. 3, pp. 253 – 264, 2002.

[79] L. Cohen, "Generalization of the Gram-Charlier/Edgeworth Series and Application to Time-frequency Analysis," *Multidimensional Systems and Signal Processing*, vol. 9, no. 4, pp. 363–372, October 1998.

[80] Harald Cramér, *Mathematical Methods of Statistics*, 2nd ed. Princeton: Princeton University Press, 1946, ch. 15.8.

[81] UCL/VINT/LBNL. Network Simulator ns (version 2). [Online]. Available: http://www.isi.edu/nsnam/ns/

[82] M. G. Rabbat, M. J. Coates, and R. D. Nowak, "Multiple-Source Internet Tomography," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2221–2234, 2006.

[83] N. Johnson, J. Thompson, S. McLaughlin, and F. Garcia, "A Comparison of Delay Estimation & Bottleneck-link Detection Methods for Network Tomography," in *Proceedings of the 8th IMA conference on Mathematics in Signal Processing*, Cirencester, UK, Dec 2008.

[84] I. Csiszár and P. C. Shields, Eds., *Information Theory and Statistics: A Tutorial*. now Publishers Inc., 2004, vol. 1, no. 4.

[85] D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing," *Commun. ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[86] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous Connections and Onion Routing," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, May 1998.

[87] S. J. Murdoch and G. Danezis, "Low-cost Traffic Analysis of Tor," in *Proc. IEEE Symposium on Security and Privacy*, 8–11 May 2005, pp. 183–195.

[88] J.-F. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, H. Federrath, Ed., vol. 2009. Springer, 2000, pp. 10–29.

[89] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On Flow Correlation Attacks and Countermeasures in Mix Networks," *Lecture Notes in Computer Science*, vol. 3424, pp. 207–225, 2005.

[90] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing Attacks in Low-latency Mix Systems," in *Proceedings of the 8th International Financial Cryptography Conference (FC 2004), Key West, FL, USA, February 2004, volume 3110 of Lecture Notes in Computer Science*. Springer, 2004, pp. 251–265.

[91] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an Analysis of Onion Routing Security," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, H. Federrath, Ed., vol. 2009. Springer, 2000, pp. 96–114.

[92] A. Serjantov and G. Danezis, *Towards an Information Theoretic Metric for Anonymity*, 2003, pp. 259–263. [Online]. Available: http://dx.doi.org/10.1007/3-540-36467-6_4

[93] LBNL Network Research Group. tcpdump. LBNL. [Online]. Available: http://www.tcpdump.org/

[94] ——. libpcap. LBNL. [Online]. Available: http://ee.lbl.gov

[95] F. Keil. privoxy. [Online]. Available: http://www.privoxy.org/