RICE UNIVERSITY

Multiscale Queuing Analysis, Sampling Theory, and Network Probing

by

Vinay Ribeiro

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

Richard G. Baraniuk, Chair, Professor
Electrical and Computer Engineering

Rudolf H. Riedi, Assistant Professor
Statistics

Edward W. Knightly, Associate Professor
Electrical and Computer Engineering

Robert Nowak, Associate Professor
Electrical and Computer Engineering
(University of Wisconsin-Madison)

Peter Druschel, Professor
Computer Science

Houston, Texas
February, 2005
ABSTRACT

Multiscale Queuing Analysis, Sampling Theory, and Network Probing

by

Vinay Ribeiro

This thesis develops novel multiscale solutions for problems in three fields – queuing theory, sampling theory, and network inference. First, we study the tail probability of an infinite-buffer queue fed with an arbitrary traffic source. The tail probability is a critical quantity for the design of computer networks. We propose a multiscale framework that uses traffic statistics at only a fixed finite set of time scales and derive three approximations for the tail probability. Theory and simulations strongly support the use of our approximations in different networking applications. Second, we design strategies to optimally sample a process in order to estimate its global average. Our results have implications for Internet measurement, sensor network design, environmental monitoring, etc. We restrict our analysis to linear estimation of certain multiscale stochastic processes – independent innovations trees and covariance trees. Our results demonstrate that the optimal solution depends strongly on the correlation structure of the tree. We also present an efficient “water-filling” solution for arbitrary independent innovations trees. Third, we present two probing tools that estimate the available bandwidth of network paths and locate links with scarce bandwidth. These tools aid network operations and network-aware applications such as grid computing. We use novel packet trains called “chirps” that simultaneously probe the network at multiple bit-rates which im-
proves the efficiency of the tools. We validate the tools through simulations and Internet experiments.
Acknowledgments

I have many people to thank for making this thesis possible. First and foremost I must praise the Almighty God whom I must thank for every good thing. I especially thank Richard Baraniuk and Rolf Riedi for the tremendous effort they have put in to guide my research during the past few years. Several other people have mentored me during graduate school including Rob Nowak, Peter Druschel, and Edward Knightly who have kindly agreed to be on my thesis committee, Walter Willinger, Anja Feldmann, Sue Moon, Zhi-Li Zhang, Christophe Diot, Anna Gilbert, Matt Grossglauser, and Nick Duffield who supervised my internships at AT&T Labs and Sprint ATL.

I am indebted to my parents for their support and advice during my entire lifetime. I specially thank my brother Rahul for his visits to Houston from Aggieland that made many of my weekends enjoyable.

I have valued the company of my colleagues and friends at Rice. Special thanks goes to Matt Crouse, Justin Romberg, Shriram Sarvotham, Ramesh Neelamani, Vishwas Sundaramurthy, Felix Fernandes, Dinesh Rajan, Sripiya Venkataraman, William Mantzel, Mike Wakin, Kadim Tasdemir, Tao Wu, and Benjamin Levy. The staff at Rice deserve a word of appreciation for their tireless work “behind the scenes.” These include Adria Baker and others at the Office of International Students, Liz Hickman, Myrl Carlson, Michael Dye, Bea Sparks and Mandy Nevin.

I would not have made it through graduate school without the aid of my spiritual mentors Jim Tour, Fr. Jack Solarski and J. L. Marti and other Opus Dei members. I wish to particularly thank campus ministers Mike Karim, Trae Vacek, C. J. Fretheim,
Fr. Mark, Fr. Mike Gray, and Sr. Kitty Carter for their support. Swaminathan Nagara, his wife Meena, and George Mathew were a source of strength during my stay in California. My friends at Bible study were like my brothers and sisters. These include Yi Wan, Brent Hendricks, Musie Ghebremichael, Wakaba Futamura, Christie Peebles, Luke Probst, William Chan, Luke Chuang, Sujesh Sreedharan, Erion and Amarda Plakue, and Chris Ralston.

I thank Kjell-Ove Widman, Ingemar Kaj and Gunnar Karlsson for giving me an opportunity to work at Institut Mittag-Leffler in beautiful Djursholm. My appreciation for the Opus Dei folks at Larkstadens, especially Denis Searby, cannot be enough. I thank my roommates Markus Fidler and Bernardo D’Auria for their company. I wish to thank Robert Hsieh, Antonio Nucci, and Michael & Lisa Liberty for making my stint in California a pleasure.

Last but not least I thank Terry Gaschen for imparting some of his classical guitar wizardry to me at Shepherd School (Rice) and Cullen Hall (University of St. Thomas).
Contents

Abstract ii
Acknowledgments iv
List of Illustrations xi
List of Tables xvii

1 Introduction 1
1.1 The success of multiscale techniques . . . . . . . . . . . . . . . . . . . 1
1.2 Queuing analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
1.3 Sampling theory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
1.4 Network probing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
1.5 Outline of thesis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

2 Queuing Analysis 17
2.1 Review of critical timescale analysis . . . . . . . . . . . . . . . . . . . . 18
2.1.1 Queue size as a multiscale function . . . . . . . . . . . . . . . . 18
2.1.2 Critical timescale queuing approximation . . . . . . . . . . . . . 19
2.2 Multiscale queuing approximations . . . . . . . . . . . . . . . . . . . 20
2.2.1 Max approximation . . . . . . . . . . . . . . . . . . . . . . . . . 20
2.2.2 Product and sum approximations . . . . . . . . . . . . . . . . . 22
2.2.3 Intuition for the accuracy of the approximations . . . . . . . . . 22
2.3 Traffic models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
3 Sampling Theory 50

3.1 Multiscale stochastic processes ........................................ 50
  3.1.1 Terminology and notation .......................................... 51
  3.1.2 Covariance trees ...................................................... 52
  3.1.3 Independent innovations trees ...................................... 53
3.2 Optimal leaf sets for independent innovations trees ............. 54
  3.2.1 Water-filling .......................................................... 54
  3.2.2 Optimal leaf sets through recursive water-filling ............. 56
  3.2.3 Uniform leaf nodes are optimal for scale-invariant trees .... 59
3.3 Covariance trees .......................................................... 60
  3.3.1 Problem formulation ............................................... 60
  3.3.2 Optimal solutions ................................................... 60
  3.3.3 Worst case solutions ................................................. 63
3.4 Numerical results .......................................................... 64
  3.4.1 Independent innovations trees: scale-recursive water-filling .. 64
  3.4.2 Covariance trees: optimal and worst cases ...................... 66
3.5 Related work .............................................................. 67
3.6 Summary ................................................................. 69

4 Network Probing 70

4.1 Network model and terminology ........................................ 70
4.2 pathChirp ................................................................. 71
  4.2.1 Algorithm ............................................................ 71
  4.2.2 Implementation details ............................................ 76
  4.2.3 Performance and parameter choice ............................... 77
4.2.4 Comparison with TOPP ........................................... 82
4.2.5 Comparison with pathload ......................................... 85
4.2.6 Internet experiments .................................................. 88
4.3 STAB ................................................................. 90
  4.3.1 Algorithm .......................................................... 90
  4.3.2 Related work on thin link localization ......................... 92
  4.3.3 Validation of STAB through simulations ...................... 93
  4.3.4 STAB Internet experiment ....................................... 97
4.4 Summary .............................................................. 100

5 Future Directions ......................................................... 102
  5.1 Queuing analysis ...................................................... 102
  5.2 Sampling theory ..................................................... 103
  5.3 Network probing ..................................................... 104

A Proofs of queues with Gaussian traffic as input .................. 105

B Proofs of WIG and MWM fed queues .................................. 115

C Proofs for independent innovations trees ............................. 122

D Proofs for covariance trees ............................................... 132

E Pseudo-code and computational complexity of water-filling al-
  gorithm ................................................................. 137

F Pseudo-code for pathChirp algorithm .................................. 143
Illustrations

1.1 The Internet. Data is forwarded from one router to the next in the form of packets. Queues buffer incoming bursts of packets.

1.2 (a) Comparison of the max, the product, and the sum approximations to \( P \{ Q > b \} \) for a queue fed with (a) fractional Gaussian noise (fGn) traffic and (b) multifractal wavelet model (MWM) traffic. For both traffic models the product and sum approximations are close to \( P \{ Q > b \} \) for a wide range of queue thresholds \( b \). The max approximation is a lower bound of \( P \{ Q > b \} \) and is accurate to an order of magnitude.

1.3 Multiscale quad-tree model of a spatial physical process. Nodes at lower levels in the tree correspond to averages of a physical process over smaller rectangular areas.

1.4 Optimal leaf sets of different sizes for two different independent innovations trees: (a) tree with balanced variance of innovations, (b) tree with unbalanced variance of innovations. In (a) the uniform leaf node sets are optimal whereas in (b) the nodes on the left half of the tree are more preferable to those on the right.

1.5 (a) Chirp packet train; the exponential flight pattern enables efficient available bandwidth estimation. (b) Packet tailgating chirp train. We replace each packet in (a) by a large packet followed closely by a small one.

1.6 Thin link localization experiment topology. The paths share eight links.
1.7  STAB’s real-time estimates of available bandwidth vs. segment length for the 
(a) UWisc – Rice and (b) UIUC – Rice paths corresponding to the topology 
depicted in Fig. 1.6. In (a) observe a steep drop at link 13 and in (b) at link 14 
indicating that they are thin links.

2.1  (a) Multiscale tree representation of a traffic trace. Nodes at each horizontal 
level in the tree correspond to the sum (aggregates) of the process in 
non-overlapping blocks of sizes of powers of two, with lower levels 
conforming to smaller block sizes. Each node is the sum of its two child 
nodes. (b) The WIG model generates the sibling nodes $V_{j+1,2k}$ and $V_{j+1,2k+1}$ 
as the sum and difference of the parent node $V_{j,k}$ and a random Gaussian 
innovation $Z_{j,k}$. (c) The MWM model generates the multiplier 
$U_{j,k} \sim \beta(p_j,p_j)$, and then form the two nodes at scale $j + 1$ by multiplying 
$V_{j,k}$ with $U_{j,k}$ and $1 - U_{j,k}$.

2.2  (a) $\zeta^*(\alpha)$ vs. $\alpha$. For a large range of $\alpha$, $\zeta^*(\alpha)$ is close to 1. (b) Comparison of 
$L^{[0]}(b)$ to its upper bound $C(b)$, for different values of $\alpha$. Observe that the 
lower bound of $L^{[0]}(b)$ is almost identical to the upper bound $C(b)$.

2.3  Comparison of the max the product and the sum approximations to 
$\P\{Q > b\}$ for iGn traffic with parameters $H = 0.8$, link capacity 10Mbps and 
$\sigma = 8 \times 10^5$ bits. For different utilizations in (a) and (b), the product and sum 
approximations are close to $\P\{Q > b\}$ for a wide range of queue thresholds $b$. 
The max approximation is a lower bound of $\P\{Q > b\}$ and is accurate to an 
order of magnitude.
2.4 Histograms of the bytes-per-time processes at time-scale 2ms for (a) wide-area traffic at the University of Auckland (trace AUCK) [62], (b) one realization of the WIG model, and (c) one realization of the MWM. Note the large probability mass over negative values for the WIG model. 42

2.5 Histograms of the bytes-per-time processes at time-scale 2.77ms for (a) video traffic formed by multiplexing 15 video traces (trace VIDEO), (b) one realization of the WIG model, and (c) one realization of the MWM. Note that the MWM matches the marginal of the video traffic better than the WIG; however, the video traffic is more Gaussian than the AUCK traffic. 44

2.6 Queuing performance of real data traces and synthetic WIG and MWM traces at high utilization. In (b), we observe that the MWM synthesis matches the queuing behavior of the AUCK data closely, while in (a) the WIG synthesis is not as close. In (c) and (d), we observe that both the WIG and the MWM match the queuing behavior of VIDEO. We also observe that the product approximation \( P^{[0z]}(b) \) is close to the empirical queuing behavior for both synthetic traffic loads (both WIG and MWM) and that it performs better than the max approximation, \( L^{[0z]}(b) \). 48

2.7 Queuing performance of real data traces and synthetic WIG and MWM traces at low utilization. The MWM outperforms the WIG even more than at higher utilizations. 49

3.1 A multiscale stochastic process. Nodes at the bottom are called leaves and the topmost node the root. 52
3.2 Optimal leaf sets for three different independent innovations trees: (a) scale-invariant tree, (b) symmetric tree with unbalanced variance of innovations at scale 1, and (c) tree with missing leaves at the finest scale. Observe that the uniform leaf node sets are optimal in (a) as expected. In (b), however, the nodes on the left half of the tree are more preferable to those on the right. In (c) the solution is similar to (a) for optimal sets of size $n = 5$ or lower but changes for $n = 6$ due to the missing nodes.

3.3 Comparison of probing schemes for a WIG model with (a) negative correlation progression and (b) positive correlation progression. Observe that the clustered nodes are optimal in (a) while the uniform is optimal in (b). The uniform and the clustered sets give the worst performance in (a) and (b) respectively, which confirms our theoretical results.

4.1 A typical chirp queuing delay signature.

4.2 Normalized mean squared error vs. probe packet size $P$ for two utilizations: (a) 30% and (b) 70%. In most cases the MSE decreases with increasing packet size. The experiment used $\gamma = 1.2$.

4.3 Normalized MSE vs. spread factor $\gamma$ for two utilizations: (a) 30% and (b) 70%. The MSE decreases with decreasing $\gamma$.

4.4 Normalized MSE vs. busy period threshold $L$ for two utilizations: (a) 30% and (b) 70%. The error improves with decreasing $L$.

4.5 Normalized MSE vs. decrease factor $F$ for two utilizations: (a) 30% and (b) 70%. The error improves with increasing $F$.

4.6 Multi-hop experiment.
4.7 Performance in multi-hop experiments. The MSE in the case of both queues being loaded is comparable to that when only one is loaded implying that pathChirp is robust to multi-hop paths. 83

4.8 Comparison of pathChirp and TOPP in a single-hop scenario for two utilizations: (a) 30% and (b) 70%. Observe that pathChirp performs far better than TOPP. 85

4.9 Comparison of pathChirp and TOPP in multi-hop scenarios. In (a) the first queue has less available bandwidth than the second while in (b) the second has the least available bandwidth. Observe that pathChirp performs far better than TOPP. 86

4.10 Testbed setup at Rice University. 87

4.11 Setup for the Internet experiment. 89

4.12 (a) Available bandwidth estimates when Poisson traffic originates at Caltech. (b) Available bandwidth estimates when Poisson traffic originates at StarLight (Chicago). Observe that the pathChirp estimates fall in proportion to the introduced Poisson traffic. 90

4.13 Double web-farm topology. Web clients download data from servers. 93

4.14 (a) Actual and (b) STAB estimates of sub-path available bandwidth during the simulation with the topology depicted in Fig. 4.13. Estimates track the actual sub-path available bandwidth well including the dip at link 5 after time $t = 200s$. The color of the plot at any point represents the height, with the blue end of the spectrum representing small heights and the red larger heights. 94
4.15 Probability of different links being thin links at time instants (a) $t = 180s$ and (b) $t = 360s$. We compute the probabilities from the sub-path available bandwidth in Fig. 4.14(b). In (a) only link 2 has a high probability of being a thin link, while in (b) both links 2 and 5 have high probabilities of being thin links.

4.16 Thin link localization experiment topology. The paths share eight links.

4.17 STAB estimates of sub-path available bandwidth for the (a) UWisc – Rice and (b) UIUC – Rice paths corresponding to the topology depicted in Fig. 4.16. In (a) observe a steep drop at link 13 and in (b) at link 14 indicating that they are thin links.

4.18 Probability of different links being thin links for the UWisc – Rice path at time instants (a) $t = 10min$ and (b) $t = 20min$.

4.19 Probability of different links being thin links for the UIUC – Rice path at time instants (a) $t = 10min$ and (b) $t = 20min$. 
Tables

4.1 Efficiency comparison of pathChirp and pathload with iperf CBR cross-traffic. ................................................................. 88
4.2 Efficiency comparison of pathChirp and pathload with Poisson cross-traffic. ................................................................. 89
Chapter 1

Introduction

Multiscale techniques model and analyze phenomena at multiple scales in space or time. They have impacted research areas as diverse as material science [33], chemistry [76], biology [87], image processing [78], physics [32], and computer networking [77]. This thesis presents novel multiscale solutions for various problems which are particularly relevant to the field of computer networking. We begin by describing some key reasons for the success of multiscale techniques and then overview the specific problems we address in this thesis.

1.1 The success of multiscale techniques

Multiscale techniques appeal to the practitioner for several reasons. We highlight two of these that are particularly relevant to this thesis.

First, the dynamics of numerous phenomena occur at different scales thus making multiscale modeling and analysis a natural choice. For instance, take the dynamics of the solar system. The earth revolves around the sun on the timescale of a year while it rotates about its own axis on the timescale of a day. Another example is the dynamics of protein folding. While the timescale of vibration of covalent bonds is on the timescale of femtoseconds ($10^{-15}$), the folding time for proteins may well be on the timescale of seconds [32, 48]. Turbulence and the mass distribution in the universe are two other examples of natural phenomena with multiscale dynamics [32, 36].
In computer networks the dynamics of traffic displays multiscale characteristics. The work-hour patterns of humans cause strong periodicities in traffic volumes on the timescales of days and weeks while network protocols such as the Transmission Control Protocol (TCP) create periodicities in traffic at the sub-second timescale [89].

Second, multiscale techniques typically study data at a set of scales spaced exponentially relative to each other. We term such sets *exponential scales*. A significant advantage of exponential scales is that they can span a wide range of scales while being small in size. For example, just seven timescales, $10^{-j}s$, $j = 0, 1, \ldots, 6$, span the entire range of timescales from micro-seconds to seconds. Multiscale techniques thus efficiently capture the dynamics of phenomena at a wide range of scales which results in fast algorithms to solve various problems. A classic example of a fast multiscale algorithm is the wavelet transform. The wavelet transform of an $N$-point data has computational cost of the order of $N$, much smaller than the $N \log(N)$ cost of the Fast Fourier Transform (FFT) [17].

In this thesis we make generous use of multiscale techniques to address problems in queuing theory, sampling theory, and network probing which we describe next.

### 1.2 Queuing analysis

Queuing occurs in systems that possess limited resources to serve an arriving input stream of customers. When customers arrive faster than they can be served, the system temporarily buffers them in queues. Our daily experience of queues occurs at traffic stop lights, check-out registers at stores, booths for entrance tickets, and while boarding airplanes. Factories, water reservoirs at dams, and the human body (for example, energy storage in the body) are other examples of systems where queuing occurs.

Queuing is intrinsic to computer networks and intimately related to network perfor-
Networks such as the Internet transmit data in the form of packets (see Figure 1.1). Each router in the network forwards an incoming packet to an appropriate output link based on the address of its final destination. In this way packets hop from router-to-router until they reach their destination, analogous to a letter passing from one post office to the next in a postal system. Routers typically buffer incoming packets in queues where they await their turn for transmission on output links. Large bursts of incoming packets cause the queues to fill up and as a result packets face queuing delays. In case queues overflow, packets are discarded, that is they are dropped.

Maintaining low queuing delays is critical for the viability of real-time streaming applications for telephony, telemedicine, videoconferencing, economic transactions, gaming etc. Two methods of reducing queuing delays are network provisioning and admission control [15,38]. Network provisioning addresses the problem by ensuring that links have enough bandwidth to quickly transmit packets, thus reducing their waiting time in queues. Admission control addresses the problem by controlling the traffic input to the queues – computers are allowed to introduce packets into the network only if these are unlikely to fill up queues.

Both provisioning and admission control require appropriate queuing analysis tools.
Queuing analysis predicts queuing delays based on the statistics of input traffic and bandwidth resources that are available at router links. It is a rich field with a history that spans several decades [47]. The queuing analysis of classical Poisson and Markov models is particularly well-developed. However, the queuing analysis of strongly correlated processes such as those with long-range-dependence (LRD) is relatively nascent.

Many studies have proved the ubiquitous presence of LRD in a variety of traffic types including local-area network, wide-area network, and video traffic [13, 26, 50, 67]. Classical Poisson and Markov queuing techniques are unsuitable for LRD traffic which creates the need for new analytical tools. Up to now exact formulas for the queuing delay of LRD processes, other than for asymptotically large delays [30, 44, 63], have not been found and we are thus forced to use approximations.

This thesis introduces a new multiscale framework for queuing analysis that provides practical approximations for the tail probability of a queue with constant service rate. While it applies to any traffic process, it is particularly apt for LRD processes. Consider a queue with constant service rate $c$ and denote the total traffic arriving at the queue in time interval $[-t, 0]$ as $K[t]$. Note that $K[t]$ is the traffic at timescale $t$. The queue size, $Q$, at time $0$ equals the difference between the total traffic that entered the queue since it was last empty and the total traffic served since then. This fact is captured by Reich’s formula

$$Q = \sup_{t \geq 0} (K[t] - ct).$$  \hspace{1cm} (1.1)

From (1.1) we see that the queue size is a function of the traffic at different timescales.

One popular approach to approximating the tail queue probability, $\mathbb{P} \{Q > b\}$, uses the marginal distribution of traffic at a single timescale, the critical timescale, $\lambda(b)$ [61]. Intuitively the critical timescale is the most likely time for an empty queue to fill up
beyond size $b$. The critical timescale approximation is

$$\mathbb{P}\{K[\lambda(b)] - c\lambda(b) > b\} \approx \mathbb{P}\{Q > b\}. \quad (1.2)$$

The critical timescale is a function of the queue threshold $b$ as well as the traffic statistics and can take a wide range of values. For an arbitrary process computing the critical timescale requires the statistics at all timescales. Thus while the critical timescale is a powerful tool that has advanced the state-of-the-art in queuing theory, using it in practice is not straightforward.

Our multiscale approach to queuing analysis characterizes a traffic process in terms of its marginal distribution at a fixed finite set of timescales $t \in \theta$. This characterization extends the notion of the critical timescale to allow practical approximations for $\mathbb{P}\{Q > b\}$. Intuitively, we design $\theta$ such that it densely covers a wide range of timescales to ensure that at least one of its elements is close to $\lambda(b)$ for a wide range of queue sizes $b$. Replacing $\lambda(b)$ by this “closest” element of $\theta$ in (1.2) gives the max approximation for the tail queue probability. We provide two other approximations for the tail queue probability based on the marginals at timescales $\theta$, the product approximation, and the sum approximation. Some noteworthy features of our approximations are that they apply to any finite queue threshold $b$, that is, they are non-asymptotic; they apply to any traffic model including non-stationary ones; and they are simple to employ because they require traffic statistics only at few timescales $\theta$.

Our approach to analyzing a queue fed with a single source easily extends to the case where it is fed with the aggregate of several independent traffic processes. We simply convolve the marginal distributions of all the individual processes at timescales $t \in \theta$ to obtain the corresponding distributions for the aggregate process. We then apply our approximations for the tail queue probability as before.

We prove several theorems that elucidate different properties of our multiscale frame-
work for queuing. We prove that exponential time scales are optimal vis-à-vis the max approximation for a queue fed with fractional Brownian motion (fBm). We also prove numerous non-asymptotic error bounds, large-queue asymptotic results, and other bounds for the three approximations for different traffic models including fBm, fractional Gaussian noise (fGn), the wavelet-domain independent Gaussian model (WIG), and the multifractal wavelet model (MWM).

We compare the different approximations through numerical experiments. Figure 1.2 highlights some of our results. Observe that for queues fed with fGn and MWM traffic, the product and sum approximations are close to $\mathbb{P}\{Q > b\}$ for a wide range of queue thresholds $b$. The max approximation is a lower bound of $\mathbb{P}\{Q > b\}$ and is accurate to an order of magnitude.

In summary, theoretical and simulation results strongly support the use of our multi-
scale queuing analysis for different network applications including delay-sensitive provisioning and admission control. The analysis applies to arbitrary traffic processes and the queuing approximations are accurate and easy to employ. In addition, our novel approach of using traffic statistics at only a small set of timescales can potentially help develop new analysis tools for more general queuing systems than those we consider in this thesis.

1.3 Sampling theory

Sampling consists of selecting some part of a population to observe in order to estimate something about the whole population [84]. It becomes unavoidable when the cost of collecting data from the entire population is prohibitive. Sampling is critical to a number of fields including politics (election exit polls), biostatistics (design of drug testing to combat disease), marketing (sampling customer profiles), environmental monitoring (sampling pollution levels), agriculture (insect population density), Internet measurement (sampling traffic loads), signal processing (sparse sampling of signals), and sensor networks (sensor placement) [18, 19, 68, 84, 85, 88].

The accuracy of an estimate based on a particular sample set is directly related to how representative the set is of the whole population. For example, an election exit poll that samples people from all income groups is likely to give better results than one that samples only a single income group. Ideally we would like to use an optimal sample set, that is a sample set that provides the best possible estimate according to some optimality criterion.

In this thesis we design strategies to optimally sample multiscale stochastic processes. A multiscale process consists of a set of univariate random variables that are organized like the nodes of a tree (see Figure 1.3) [90]. Nodes at the bottom are called
leaves and the topmost node is the root. We associate each node with the average of a physical process over some region with nodes at higher scales corresponding to larger regions. The root thus represents the global average of the process and the leaves represent local samples.

The question we address is: Among all possible sets of leaves of size $n$, which set gives the best linear estimate of the root in terms of minimum mean squared error? Equivalently, at which locations must we strategically sample the process in order to estimate its global average optimally? This question has implications for Internet measurement, sensor network design, environmental monitoring, etc. When designing a sensor network to measure pollution, for instance, we may wish to know where to place a limited number of sensors in order to optimally estimate the total pollution in a given geographical region. Each sensor gives us a local measurement of pollution, that is a sample of the process that we model by a leaf of the tree. Our goal is to estimate the total pollution represented by the root (see Figure 1.3).

We prove optimality results for two types of multiscale processes: independent innovations trees and covariance trees. In an independent innovations tree every node is related to the node immediately above it through an independent random innovation. In
a covariance tree the covariance of leaf nodes is a function only of their distance in the tree.

For any independent innovations tree we show that the optimal solution is readily given by a polynomial-time algorithm that we term the *water-filling algorithm*. Note that the general problem of determining the set of \( n \) random variables that provide the best linear estimate of another random variable is NP-hard [27]. In contrast, our setting is one example of this general problem that is solved in polynomial-time by the water-filling algorithm.

Using the water-filling algorithm we demonstrate that the optimal leaf sets can vary from one independent tree to another. For a tree in which the variance of innovations is identical on the left and right side of the tree (see Figure 1.4(a)) the optimal sets consist of leaves uniformly spread out in the tree. For a tree in which the variance of innovations is different on the left and right side of the tree (see Figure 1.4(b)), the optimal leaf sets are concentrated toward one side.

For covariance trees we demonstrate that the optimal solutions vary dramatically with the correlation structure. For covariance trees with positive correlation progression through scales, sets of *uniformly spaced leaves* (leaves uniformly spread out in the tree) are optimal and sets of *clustered leaves* (sets of contiguous leaves) are the worst possible. For processes with negative correlation progression, sets of uniformly spaced leaves are the worst possible.

The intuition for these results is that sets of leaf nodes that are “more correlated” give worse estimates of the root. Essentially leaf nodes that are highly correlated contain more redundant information and hence cannot give good estimates of the root. In the case of trees with positive correlation progression, clustered leaf nodes are strongly correlated when compared to uniform leaf nodes. The opposite is true in the negative
Figure 1.4: Optimal leaf sets of different sizes for two different independent innovations trees: (a) tree with balanced variance of innovations, (b) tree with unbalanced variance of innovations. In (a) the uniform leaf node sets are optimal whereas in (b) the nodes on the left half of the tree are more preferable to those on the right.

correlation progression case.

In summary, the water-filling algorithm provides a fast polynomial-time algorithm to determine the optimal solution of any independent innovations tree. Our results for covariance trees prove that the optimal results can vary drastically depending on the covariance structure of the modeled process. These findings impact a wide range of applications including the design of sensor networks and Internet inference schemes.

1.4 Network probing

The Internet has become a vital component of international commerce, interpersonal communication, and technological development. Thanks to rapid technological improvements and its decentralized nature, the Internet has grown rapidly.

In order to maintain the Internet’s future growth and viability it is critical to measure
and improve its performance. Researchers, service providers, and other Internet users need to understand the Internet’s growth characteristics and its limitations, both globally and locally, in order to improve the design of the Internet and efficiently use its resources [16].

Unfortunately measuring different properties of the Internet is non-trivial. First, the Internet is owned by different entities making the sharing and dissemination of information difficult. Second, because the Internet is extremely large, any widespread measurement infrastructure would require huge resources in terms of storage memory and other equipment. Third, the measurement process by itself can degrade network performance. For example, collecting information from routers may use up precious CPU and bandwidth resources which can affect their packet forwarding performance.

One solution to the problem of Internet measurement is edge-based probing. Rather than measure internal properties of the Internet directly, edge-based probing techniques infer them from the end-to-end delay of injected probe packets. Several probing techniques have been suggested to measure the bandwidth of links on a path, the cross-traffic in simplified network settings, the delay and packet loss at individual links in the network, and the topology of the network [19, 23, 29, 31, 42, 71].

In this thesis we focus on the inference of available bandwidth. We define the available bandwidth of a link as its unused bandwidth in a given time interval. The available bandwidth of a network path is simply the least available bandwidth of all links that constitute the path. Knowledge about the available bandwidth of different paths can aid route selection, server selection, rate-based streaming applications, and network-aware applications such as grid computing [11, 22, 41].

The principle of self induced congestion provides one effective way to estimate available bandwidth [45, 59]. This principle relies on the fact that routers buffer incoming
packets in queues before transmitting them on output links (see Figure 1.1). If the incoming packet bit rate exceeds the transmission rate of the outgoing link then packets fill up the corresponding queue and face queuing delays. According to the principle of self-induced congestion if we inject probe packets into a path at a bit rate faster than the available bandwidth then the path’s queues will congest leading to increasing delays. The path’s queues will not congest, however, if the probing bit rate is less than the available bandwidth. We can thus infer the available bandwidth by injecting probe packets into the network at different bit rates and finding the minimum bit rate at which we start to see increasing delays of probe packets.

While estimating available bandwidth we are confronted with the dilemma of a probing “uncertainty principle.” By introducing more probe packets into the network we can probe at more bit rates and hence estimate available bandwidth more accurately. However, we simultaneously starve other traffic on the network of precious bandwidth resources. We thus have the task of scanning the bandwidth spectrum efficiently to obtain an accurate estimate of available bandwidth. This is particularly challenging because the available bandwidth can vary by several orders of magnitude due to the heterogeneity of Internet links. The bandwidth of a modem line can be as low as $56 \times 10^3 \text{ bits-per-second (bps)}$ and that of backbone links can be as high as $10 \times 10^9 \text{ bps}$.

Our first probing tool, pathChirp, uses a multiscale probing strategy to efficiently estimate available bandwidth [74]. It uses special packet trains called chirps in which the time interval between successive packets decreases exponentially. Because the probing bit rate is inversely proportional to the time interval between probe packets, the probing bit rate within a chirp increases exponentially fast (see Figure 1.5). Thus chirps sweep through a wide range of probing rates using a few packets which allows us to estimate the available bandwidth efficiently using the self-induced congestion principle.
We demonstrate pathChirp’s efficiency by comparing it to other probing tools through simulations and Internet experiments.

In addition to estimating the magnitude of available bandwidth of a path we also estimate spatial information about the available bandwidth, namely the location of thin links. We define a thin link as any link with less available bandwidth than all links preceding it on a path. We term the last thin link of a path the tight link. The tight link by definition has the least available bandwidth among all links of the path. Information about the locations of thin links assists network monitoring and troubleshooting and can provide insight into the causes of network congestion and ways of circumventing it.

Our second probing tool, the Spatio-Temporal Available Bandwidth estimator STAB, locates thin links. In addition to the principle of self-induced congestion, STAB employs the concept of packet tailgating [73]. Packet tailgating uses special packet-pairs; each pair consists of a large packet closely followed by a small one. These help estimate the available bandwidth of different segments of the path which extend from the source to different intermediate links.
Figure 1.6: Thin link localization experiment topology. The paths share eight links.

Figure 1.7: STAB’s real-time estimates of available bandwidth vs. segment length for the (a) UWisc – Rice and (b) UIUC – Rice paths corresponding to the topology depicted in Fig. 1.6. In (a) observe a steep drop at link 13 and in (b) at link 14 indicating that they are thin links.

STAB locates thin links from a plot of available bandwidth against segment length. This plot ideally decreases at every point corresponding to a thin link and remains constant between consecutive thin links. We thus estimate thin link locations as those points in this plot where we observe a noticeable decrease in available bandwidth.

We use STAB to locate the thin links on two paths depicted in Figure 1.6. From the plot of available bandwidth against segment length for the UWisc – Rice path in Figure
1.7(a) we observe that the available bandwidth drops sharply at link 13. This indicates that link 13 is a thin link. The plot also indicates that link 13 is the last thin link of the path, the tight link. From Figure 1.7(b) we observe that link 14 is the tight link for the UIUC – Rice path. Note from Figure 1.6 that link 13 of the UWisc – Rice path and link 14 of the UIUC – Rice path are identical. Thus STAB reveals that both paths have the same tight link.

In practice the dynamics of cross-traffic on the path introduce noise that affect our estimates of available bandwidth and locations of thin links. PathChirp and STAB address the issue of noise using different algorithms.

In summary, by using packet chirps, the principle of self-induced congestion and the concept of packet tailgating, our tools pathChirp and STAB efficiently estimate the available bandwidth of paths and locate thin links. Simulations and Internet experiments support their use for different applications. Both tools are available as freeware on the web [1].

1.5 Outline of thesis

The rest of the thesis is organized as follows. We present our contributions to the three fields of queuing analysis, sampling theory, and network probing in Chapters 2, 3, and 4, respectively. Each of these chapters ends with a summary of our contributions to the corresponding field. We conclude with some insights about future directions for this research in Chapter 5.

The reader must bear in mind one issue regarding notation while perusing this thesis. Because Chapters 2 and 3 each use a large body of symbolic notation we cannot completely avoid using the same symbols to represent different quantities in these chapters. There is no reuse of symbols within any chapter, however, and each of Chapters 2, 3,
and 4 can be read independent of the others.
Chapter 2

Queuing Analysis

In this chapter develop a new approach to queuing analysis called the \textit{multiscale queuing analysis}. We model a router queue as an infinite length queue with constant service rate [64] and study the probability that the queue size $Q$ exceeds a threshold $b$, $\mathbb{P}\{Q > b\}$, also called the \textit{tail queue probability}.

We can predict $\mathbb{P}\{Q > b\}$ in several ways. First, we can model network traffic using different processes (also called traffic models) and use any exact formula for $\mathbb{P}\{Q > b\}$ that is available. Second, in case exact results are unavailable for a particular process we can employ analytical results that only approximate $\mathbb{P}\{Q > b\}$, which we call \textit{queuing approximations}. Third, if modeling traffic with a standard random process is cumbersome or inadequate then we can predict $\mathbb{P}\{Q > b\}$ directly from measured traffic statistics. In such a scenario it is desirable to use a small number of traffic statistics in order to reduce data acquisition and computational requirements. Our analysis addresses the second and third scenarios mentioned above.

The rest of this chapter is organized as follows. We review previous work on the concept of the critical timescale, which is crucial to our analysis, in Section 2.1. In Section 2.2 we present the multiscale queuing analysis and derive various queuing approximations. Section 2.3 describes the fBm, the WIG, and the MWM traffic models. In Section 2.4 we prove the optimality of exponential timescales for fBm. Section 2.5 proves large buffer asymptotic results and Section 2.6 proves bounding results for the different queuing approximations. In Section 2.7 we prove the convergence of the product approxima-
tion. Section 2.8 demonstrates the accuracy of the approximations through simulations with Internet and synthetic model data and also demonstrates the impact of marginals on queuing. We summarize our findings in Section 2.9.

2.1 Review of critical timescale analysis

In this section we review previous work on the critical timescale queuing analysis to set the stage for our multiscale queuing analysis in subsequent sections.

2.1.1 Queue size as a multiscale function

Consider a continuous-time fluid queue with constant service rate \( c \) with traffic process \( X_\tau, \tau \in \mathbb{R} \) as input. We refer to

\[
K_\tau[t] := \int_{\tau-t}^{\tau} X_\omega \, d\omega
\]  

(2.1)

as the traffic process at timescale \( t \). To avoid notational ambiguity we occasionally add superscripts such as in \( K^{(X)}_\tau[t] \) to identify the traffic process. For the ease of notation we drop the subscript \( \tau \) for all time-invariant quantities.

Assuming that the queue was empty at some time instant prior to \( \tau \), the queue size \( Q_\tau \) equals the difference between the total traffic that arrived at the queue and the total traffic serviced since the time instant the queue was last empty. This is succinctly captured by Reich’s formula [70]

\[
Q_\tau := \sup_{t>0} (K_\tau[t] - ct).
\]  

(2.2)

We address the requirement of an empty queue prior to \( \tau \) with mathematical rigor in Section 2.3.5.

A key interpretation of (2.2) is that \( Q_\tau \) equals a function of \( K_\tau[t] \), the traffic process
at all timescales \( t \). The question arises as to whether or not we can accurately approximate \( \mathbb{P}\{Q > b\} \) using the distribution of \( K_\tau[t] \) at a single timescale \( t \).

### 2.1.2 Critical timescale queuing approximation

Most proposed approximations of \( \mathbb{P}\{Q > b\} \) for queues fed by LRD traffic are indeed based on a single timescale called the critical timescale [20, 30, 34, 39, 61, 63]

\[
\lambda_\tau(b) := \arg \sup_{t > 0} \mathbb{P}\{K_\tau[t] - ct > b\}.
\]

(2.3)

We term the associated queue tail approximation the critical timescale approximation

\[
C_\tau(b) := \sup_{t > 0} \mathbb{P}\{K_\tau[t] - ct > b\} = \mathbb{P}\{K_\tau[\lambda_\tau(b)] - c\lambda_\tau(b) > b\}.
\]

(2.4)

Clearly \( C_\tau(b) \) is a lower bound of \( \mathbb{P}\{Q_\tau > b\} \) since by (2.2) \( K_\tau[\lambda_\tau(b)] - c\lambda_\tau(b) \leq Q_\tau \); thus

\[
C_\tau(b) \leq \mathbb{P}\{Q_\tau > b\}.
\]

(2.5)

Earlier work based on large deviation theory has shown that \( C_\tau(b) \) has the same log-asymptotic decay as \( \mathbb{P}\{Q_\tau > b\} \) when \( b \to \infty \) for a large class of input traffic processes including fBm [30, 63]. As the simulations in Section 2.8 demonstrate, \( C_\tau(b) \) is also a good approximation for \( \mathbb{P}\{Q_\tau > b\} \) for any finite \( b \) for fBm-fed queues. The intuition for the accuracy of \( C_\tau(b) \) is that “rare events occur in the most likely way.” In other words given that \( \{Q_\tau > b\} \) is a rare event, if the queue size is conditioned to fill up greater than \( b \) then it does so in time \( \lambda_\tau(b) \) in which this is most likely. That is, conditioned on \( \{Q_\tau > b\} \), we have that \( Q_\tau \) is approximately equal to \( K[\lambda_\tau(b)] - c\lambda_\tau(b) \).

While the critical timescale is a powerful tool that has advanced the state-of-the-art in queuing theory, using it in practice is not straightforward. First, consider the problem of computing \( C_\tau(b) \) for a queue fed with an arbitrary process, solely from empirical
traffic measurements. From (2.4) we see that we require the distribution of $K_\tau[t]$ for all possible $t$. This is infeasible to obtain empirically. Even if we replace purely empirical schemes by techniques that use both empirical statistics and analytical models, similar computational problems may persist. For example if we use traffic models for which analytical expressions for $C_\tau(b)$ are unknown then we may have to employ computationally intensive algorithms to determine $C_\tau(b)$. Second, say that we wish to compute the critical timescale approximation when two independent processes $X$ and $Y$ are multiplexed and input to a queue. Such a scenario often arises in admission control and network provisioning [15, 38]. Obtaining $C_{\tau}^{\{X+Y\}}(b)$ directly from the statistics of $X$ and $Y$ is again fraught with similar problems.

2.2 Multiscale queuing approximations

In this section we develop three new queuing approximations that do not have the computational problems that are associated with using the critical timescale approximation. A key factor that simplifies their computation is that they use traffic statistics only at a fixed finite set of timescales $\theta \subset \mathbb{R}_+$. Note that while some of our theoretical results are for countably infinite sets $\theta$, in practice we always employ a truncated, finite set $\theta$ when computing the queuing approximations. We typically choose the set $\theta$ to span the range of timescales in which we expect the critical timescale $\lambda(b)$ to lie, for values of $b$ relevant to a particular application.

2.2.1 Max approximation

In analogy to the queue size formula and the critical timescale (see (2.2) and (2.3)) define

$$Q_\tau^{[\theta]} := \sup_{t \in \theta} (K_\tau[t] - ct) \quad (2.6)$$
and

$$\lambda[\theta]^\tau(b) := \arg \sup_{t \in \theta} \mathbb{P}\{K_\tau[t] - ct > b\}$$  \hspace{1cm} (2.7)

for $\theta \subset \mathbb{R}_+$. This leads to the max approximation

$$L[\theta]^\tau(b) := \sup_{t \in \theta} \mathbb{P}\{K_\tau[t] - ct > b\} = \mathbb{P}\left\{K_\tau\left[\lambda[\theta]^\tau\right] - c\lambda[\theta]^\tau > b\right\}. \hspace{1cm} (2.8)$$

Comparing (2.4) to (2.8) we see that the max approximation is similar to the critical timescale approximation with the difference that the supremum is taken over a finite set in (2.8) instead of over all timescales in (2.4). From (2.4), (2.5), and (2.8) we have the bounds

$$L[\theta]^\tau(b) \leq C_\tau(b) \leq \mathbb{P}\{Q_\tau > b\}.$$  \hspace{1cm} (2.9)

We note from (2.2) and (2.6) that

$$Q_\tau = Q[\tau]^\mathbb{R}_+ \geq Q[\tau]^\theta$$  \hspace{1cm} (2.10)

and from (2.6), (2.8), and (2.10) that

$$L[\theta]^\tau(b) \leq \mathbb{P}\{Q[\theta]^\tau > b\} \leq \mathbb{P}\{Q_\tau > b\}. \hspace{1cm} (2.11)$$

The max approximation is a practical replacement for $C_\tau(b)$. Since the max approximation requires estimates of $\mathbb{P}\{K_\tau[t] - ct > b\}$ only for $t \in \theta$, the difficulties associated with computing $C_\tau(b)$ as we described earlier do not arise. First, consider the problem of obtaining the max approximation from empirical traffic measurements. We simply compute histograms of the traffic at timescales $t \in \theta$ and then estimate $\mathbb{P}\{K_\tau[t] - ct > b\}$. Second, consider the problem of computing the max approximation when two independent processes $X$ and $Y$ are multiplexed and input to a queue. By simply convolving the distributions of $K_\tau^X[t]$ and $K_\tau^Y[t]$ for $t \in \theta$ we obtain the corresponding distributions of $K_\tau^{X+Y}[t]$, which immediately give the max approximation.
2.2.2 Product and sum approximations

Two additional approximations of $\mathbb{P}\{Q_\tau > b\}$ based on the set of timescales $\theta$ are the product approximation

$$P^{[\theta]}_p(b) := 1 - \prod_{t \in \theta} \mathbb{P}\{K_\tau[t] - ct < b\}$$

(2.12)

and the sum approximation

$$S^{[\theta]}_s(b) := \sum_{t \in \theta} \mathbb{P}\{K_\tau[t] - ct > b\}.$$  

(2.13)

Note that the product approximation equals $\mathbb{P}\{Q^{[\theta]}_\tau > b\}$ if the events $\{K_\tau[t] - ct > b\}$, $t \in \theta$, are independent,* and that the sum approximation equals $\mathbb{P}\{Q^{[\theta]}_\tau > b\}$ if the same events are mutually exclusive.

2.2.3 Intuition for the accuracy of the approximations

The max, product, and sum approximations inherit the accuracy of the critical timescale approximation while being practical. If there exists an element of $\theta$ close enough to the critical time-scale then $L^{[\theta]}_p(b)$ will be close to $C_\tau(b)$ (see (2.4) and (2.8)). Moreover, if a single probability term dominates the summation in (2.13), then the product and sum approximations will closely approximate $L^{[\theta]}_p(b)$ and hence $C_\tau(b)$. Simulations below in Section 2.8 demonstrate that the product and sum approximations are often closer to $\mathbb{P}\{Q_\tau > b\}$ than the max approximation. In subsequent sections we study several issues related to the three approximations.
2.3 Traffic models

This section describes four traffic models that we focus on in this chapter. While all have been shown to model the LRD in real Internet traffic well, they differ in their ability to model other properties of traffic.

2.3.1 Fractional Brownian motion

Fractional Brownian motion (fBm) is the unique Gaussian process with stationary increments and the following scaling property for all $a > 0$, $\tau \in \mathbb{R}$, and $0 < H < 1$ [55]

$$B_{a \tau} \overset{d}{=} a^H B_{\tau}.$$  \hfill (2.14)

The symbols “$\overset{d}{=}”$, “var”, $\mathbb{E}$ and “cov” denote equality in distribution, variance, expectation, and covariance respectively.

*If events $E_i$, $i \in \mathbb{N}$, are independent then so are their complements.*
2.3.2 Fractional Gaussian noise

Fractional Gaussian noise (fGn) is the increment process of fBm. While fGn is stationary, fBm is itself non-stationary by definition. Denote the stochastic differential of $B_\tau$ as $\Delta_\tau B$. We denote fGn by

$$G_\tau[t] := K_\tau^{(\Delta_\tau B)}[t] = B_\tau - B_{\tau-t}. \quad (2.15)$$

While it is difficult to define $\Delta_\tau B$ rigorously, its aggregate $K_\tau^{(\Delta_\tau B)}[t]$ is well defined. Often one is interested only in the time series $\{G_{it'}[t']\}_{i \in \mathbb{Z}}$ with $t'$ a constant time lag. From (2.14) and (2.15) we have that

$$K_\tau^{(\Delta_\tau B)}[t] \overset{d}{=} B_t \overset{d}{=} t^H B_1 \quad (2.16)$$

and thus

$$\text{var}(G_{it'}[t']) = \text{var}(K_\tau^{(\Delta_\tau B)}[t']) = \sigma^2(t')^{2H} \quad (2.17)$$

where $\sigma^2 = \text{var}(B_1)$. When $1/2 < H < 1$, fGn possesses LRD.

2.3.3 Wavelet-domain independent Gaussian (WIG) model

The WIG is a Gaussian traffic model that is able to approximate fBm and fGn as well as processes with more general scaling than (2.14) and (2.17). It uses a multiscale tree to model traffic over the time interval $[0, T]$ [53, 54]. The nodes $V_{j,k}$ on the tree correspond to the total traffic in the time interval $[k2^{-j}T, (k+1)2^{-j}T], k = 0, \ldots, 2^j - 1$ (see Fig. 2.1).

Starting at node $V_{j,k}$, the WIG models its two child nodes $V_{j+1,2k}$ and $V_{j+1,2k+1}$ using independent additive random innovations $Z_{j,k}$ through

$$V_{j+1,2k} = (V_{j,k} + Z_{j,k})/2, \quad V_{j+1,2k+1} = (V_{j,k} - Z_{j,k})/2. \quad (2.18)$$
In practice one uses a WIG tree of finite depth $n$ to obtain a discrete-time process $V_{n,k}$. The $Z_{j,k}$ have the same variance within each scale $j$, thus guaranteeing that $V_{n,k}$ is a first-order stationary process. The root $V_{0,0}$ and all $Z_{j,k}$ are Gaussian which ensures that all tree nodes are Gaussian.

To fit a traffic model means to choose its parameters either to match key statistics of observed traffic or to ensure that the model has certain prespecified statistical properties. Fitting the WIG involves choosing its parameters to obtain a required variance progression of $\text{var}(V_{j,k})$. The WIG can provide a Gaussian approximation for any stationary discrete-time process $X$; that is the WIG can be fit to obtain

$$\text{var}(V_{n-j,k}) = \text{var}\left(K^{\{X\}[2^j]}\right).$$

(2.19)

We will hitherto refer to a WIG model for which (2.19) holds as a “WIG model of $X$”. Synthetic WIG data has been shown to match the queueing behavior of Gaussian-like Internet traffic well [54].

### 2.3.4 Multifractal wavelet model (MWM)

The MWM is a non-Gaussian model based on a multiscale tree that, like the WIG, allows a more general scaling behavior of the variance of tree nodes than fGn [77]. Unlike the WIG, it ensures positivity at all time scales, an intrinsic property of real data traffic that is often ill approximated by Gaussian models. Setting $V_{0,0} \geq 0$ the MWM uses independent multiplicative innovations $U_{j,k} \in [0, 1]$ to model the two children of node $V_{j,k}$ through

$$V_{j+1,2k} := V_{j,k}U_{j,k},$$

$$V_{j+1,2k+1} := V_{j,k}(1 - U_{j,k}).$$

(2.20)

Because the product of independent random variables converges to a log-normal distribution by the central limit theorem, the nodes $V_{j,k}$ become approximately log-normal
with increasing $j$.

Following [77], we model the $U_{j,k}$’s and $V_{0,0}$ as symmetric beta random variables.

\[ U_{j,k} \sim \beta(p_j, p_j), \quad p_j \geq 0 \tag{2.21} \]

and $V_{0,0}$ as

\[ V_{0,0} \sim \rho U_{-1} \tag{2.22} \]

with $\rho \geq 0$ a constant and $U_{-1} \sim \beta(p_{-1}, p_{-1})$, an asymmetric beta random variable.

In our simulation experiments we set $p_{-1} = q_{-1}$. The tree node $V_{j,k}$ is thus the product of several independent beta random variables. Using Fan’s result [37], we approximate the distribution of $V_{j,k}$ as another beta distribution with known parameters in order to compute different queuing approximations for the MWM.

Fitting the MWM involves choosing its parameters to obtain a required variance progression of $\text{var}(V_{j,k})$. The MWM can model any stationary discrete-time process $X$ with positive autocovariance in the sense of (2.19). It has been shown to capture the queuing behavior of certain heavy-tailed, non-Gaussian traffic well [75].

While the WIG and MWM models are first-order stationary, they are not second-order stationary. This is apparent from Fig. 2.1. Observe that $V_{j+2,4k}$ and $V_{j+2,4k+1}$ have the same parent node while $V_{j+2,4k+1}$ and $V_{j+2,4k+2}$ do not. Thus the correlation of $V_{j+2,4k+1}$ with its two neighbors, $V_{j+2,4k}$ and $V_{j+2,4k+2}$, are different. Both models however have a time-averaged correlation structure that is close to the stationary process $X$ that they model (see [53, 77] for details).

### 2.3.5 Queuing analysis setup for fBm, fGn, WIG, and MWM

We now state precisely the queuing setup for the fBm, fGn, WIG, and MWM models that we analyze in subsequent sections. We set the initial queue size to be empty to satisfy the sufficient condition for (2.2) to hold (see Section 2.1.1).
All queuing results in this thesis for queues with fBm input correspond to a continuous-time queue with service rate \( c \), initial value \( Q_0 := 0 \), and \( K_\tau[t] = K_\tau^{\Delta, B+m}[t] = B_\tau - B_{\tau^{-t}} + mt \). We have

\[
P\{Q_\tau > b\} = P\left\{ \sup_{0 \leq t \leq \tau} (K_\tau[t] - ct) > b \right\}
\]

\[
\xrightarrow{\tau \to \infty} P\left\{ \sup_{t \geq 0} (K_0[t] - ct) > b \right\} =: P\{Q_\infty > b\}
\]

(2.23)

where the limit holds because of stationarity of fBm increments and Lemma 34 (in Appendix A). We assume that \( \tilde{c} := c - m > 0 \) and study the quantity \( P\{Q_\infty > b\} \) as defined in (2.23).

For fGn, WIG, and MWM traffic we consider discrete-time queues that are initialized to \( Q_0 := 0 \) and evolve according to

\[
Q_{\tau+1} = \max(Q_\tau + X_\tau - \tau, 0), \quad \tau \in \mathbb{Z}_+.
\]

(2.24)

Defining \( K_\tau[t] := \sum_{k=\tau^{-t}}^{\tau-1} X_k \) for \( t = 1, 2, \ldots, \tau \) and \( \tau = 1, 2, \ldots, \infty \), and \( K_\tau[0] := 0 \) we have

\[
Q_\tau := \max_{t=0,1,\ldots,\tau} (K_\tau[t] - \tau t).
\]

(2.25)

For fGn we set \( \tau = ct' \) and \( X_\tau = G_{\tau t'}[t'] \) for \( \tau = 0, 1, \ldots, \infty \). We study the quantity \( P\{Q_\infty > b\} \) which is defined as in (2.23) with the difference that \( t \) and \( \tau \) take integer values.

For the WIG and MWM we consider \( Q_\tau \) only for \( \tau = 0, 1, \ldots, 2^n - 1 \) with \( \bar{c} = \tilde{c}^{(n)} := cT2^{-n} \) where \( n \) is the depth of the multiscale tree. Here \( X_\tau = \mathbb{V}_{n,\tau} \). We assume that

\[
\mathbb{E}(\mathbb{V}_{n,k}) < \tilde{c}^{(n)}
\]

(2.26)

and study \( P\{Q_\tau > b\} \) which is a time-varying quantity.
For the fGn, WIG, and MWM models $K_{	au}[t]$ is only defined for $t = 0, 1, \ldots, \tau$. For these models we define $L_{\tau}^{[\theta]}(b)$, $P_{\tau}^{[\theta]}(b)$, and $S_{\tau}^{[\theta]}(b)$ as in (2.8), (2.12), and (2.13) except that we replace $\theta$ by $\theta \cap \{0, 1, \ldots, \tau\}$.

### 2.4 Optimality of exponential time scales for the max approximation of an fBm queue

Comparing (2.4) and (2.8) we see that the more dense $\theta$ is in $\mathbb{R}_+$, the closer the max approximation is to $C_{\tau}(b)$. However, we simultaneously have to acquire data at more time scales, and the max computational cost increases (see (2.8)). In this section we prove that the sets of exponential time scales

$$
\theta_\alpha := \{\alpha^k : k \in \mathbb{Z}\}, \quad \alpha > 1,
$$

optimally balances this trade-off in accuracy versus computational cost.

More precisely, for a queue with fBm input we first obtain a non-asymptotic bound on the error of $L_{\tau}^{[\theta_\alpha]}(b)$ in approximating $C(b)$. This bound proves that $L_{\tau}^{[\theta_\alpha]}(b)$ accurately approximates $C(b)$ for a wide range of $\alpha$. Second, we prove that $\theta_\alpha$ is the most sparse of all sets $\theta$ that satisfy a particular accuracy criterion for $L_{\tau}^{[\theta]}(b)$.

#### 2.4.1 Accuracy of $L_{\tau}^{[\theta_\alpha]}(b)$

Consider a queue fed by fBm traffic as described in Section 2.3.5. Then for $t > 0$, using (2.14) it is easily shown that [61]

$$
\mathbb{P}\{K[t] - ct > b\} = \Phi(g(b, t))
$$

where

$$
g(b, t) := \frac{b + \tilde{c}t}{\sigma t^H} = \frac{b + (c - m)t}{\sigma t^H}
$$
and $\Phi$ is the complementary cumulative distribution function of a zero mean unit variance Gaussian random variable [46]. From (2.4) and (2.8) we have

$$C(b) = \sup_{t > 0} \Phi(g(b, t)) = \Phi\left(\inf_{t > 0} g(b, t)\right)$$

and

$$L^{[\theta]}(b) = \sup_{t \in \theta} \Phi(g(b, t)) = \Phi\left(\inf_{t \in \theta} g(b, t)\right).$$

We characterize the accuracy of $L^{[\theta]}(b)$ in terms of the following metric

$$h_{\theta} := \sup_{b > 0} \frac{\inf_{t \in \theta} g(b, t)}{\inf_{t > 0} g(b, t)}.$$  \hspace{1cm} (2.32)

Intuitively, the closer $h_{\theta}$ is to 1 the tighter we can bound the error of $L^{[\theta]}(b)$ in approximating $C(b)$.

The following theorem states the remarkable fact that $h_{\theta}$ is solely a function of the largest ratio of consecutive time scales in $\theta$ and does not depend on any other property of $\theta$. In addition $h_{\theta}$ is not a function of the traffic model and queue parameters $m$, $\sigma$, and $c$.

**Theorem 1** Let $\theta = \{t_k\}_{k \in \mathbb{Z}}$ be a countable set of time scales such that

$$\sup_{k} t_k = \infty \quad \text{and} \quad \inf_{k} t_k = 0.$$ \hspace{1cm} (2.33)

Assuming $t_{k-1} < t_k$, denote the largest ratio of consecutive time scales by

$$d_{\theta} := \sup_{k} \frac{t_k}{t_{k-1}}.$$ \hspace{1cm} (2.34)

Assume that $d_{\theta} < \infty$, $0 < H < 1$ and $\hat{c} > 0$. Then the accuracy metric of $\theta$ is given by

$$h_{\theta} = \zeta(d_{\theta}, H) := \frac{(d_{\theta} - 1)^H (1 - H)^{1-H}}{(d_{\theta} - d_{\theta}^H)^{1-H}(d_{\theta}^H - 1)^H}.$$ \hspace{1cm} (2.35)
The proof is in Appendix A.

According to Theorem 1, to set the accuracy metric $h_\theta$ we have only to choose the largest ratio of consecutive time scales $d_\theta$ appropriately. From (2.32) and (2.34) note that $h_\theta \geq 1$ and $d_\theta > 1 \forall \theta$. In Appendix A we prove that $h_\theta$ is closer to 1 for values of $d_\theta$ closer to 1.

We can use Theorem 1 to obtain the maximum error of $L^{[\theta_\alpha]}(b)$ in approximating $C(b)$ for all possible fBm traffic processes satisfying $\widehat{c} > 0$.

**Corollary 2** For fBm input traffic with $\widehat{c} > 0$

\[
\Phi \left( \zeta^*(\alpha) \Phi^{-1}(C(b)) \right) \leq L^{[\theta_\alpha]}(b) \leq C(b),
\]

where $\zeta^*(\alpha) := \max_{H \in (0,1)} \zeta(\alpha, H)$. Equivalently

\[
L^{[\theta_\alpha]}(b) \leq C(b) \leq \Phi \left( \frac{1}{\zeta^*(\alpha)} \Phi^{-1}(L^{[\theta_\alpha]}(b)) \right).
\]

**Proof:** Note that by the construction of $\theta_\alpha$ (see (2.27)), $d_{\theta_\alpha} = \alpha$. Thus from (2.9), (2.30)–(2.32), and (2.35) we have

\[
\begin{align*}
C(b) &\geq L^{[\theta_\alpha]}(b) = \Phi \left( \inf_{t \in \theta_\alpha} g(b, t) \right) \\
&\geq \Phi \left( h_{\theta_\alpha} \inf_{t > 0} g(b, t) \right) = \Phi \left( \zeta(\alpha, H) \Phi^{-1}(C(b)) \right) \\
&\geq \Phi \left( \zeta^*(\alpha) \cdot \Phi^{-1}(C(b)) \right).
\end{align*}
\]

\[\square\]

In Fig. 2.2(a) we see that the plot of $\zeta^*(\alpha)$ versus $\alpha$, which we obtained numerically, is close to 1 for a large range of values of $\alpha$. As a result the lower bound of $L^{[\theta_\alpha]}(b)$ from (2.36) for different values of $\alpha$ is close to $C(b)$ as depicted in Fig. 2.2(b). In fact $L^{[\theta_2]}(b)$ is almost identical to $C(b)$ when $C(b) > 10^{-8}$. Thus $L^{[\theta_2]}(b)$ is for all practical purposes as accurate as $C(b)$ in approximating $\mathbb{P}\{Q_\infty > b\}$. We confirm the accuracy of $L^{[\theta_2]}(b)$ through simulations in Section 2.8.
2.4.2 Optimality of exponential time scales $\theta_\alpha$

Given a range of time scales $\mathcal{T}$, we wish to find that time-scale set which is the most sparse (i.e., has the fewest elements) in $\mathcal{T}$ while guaranteeing a certain accuracy of $L[\theta](b)$. The next theorem proves that for arbitrary $\mathcal{T}$, $\theta_\alpha$ is the most sparse of all sets $\theta$ that have accuracy metric $h_\theta$ less than a given threshold.

**Theorem 3** Let $\theta = \{t_k\}_{k \in \mathbb{Z}}$ satisfy (2.33). Let $\mathcal{A}_\mathcal{T}(\theta)$ denote the number of elements of $\theta$ that lie in a range of time scales $\mathcal{T} := (t, \bar{t})$, $0 < t < \bar{t}$. Denote by $\Gamma(\alpha)$ the set of all time-scale sets $\theta$ for which accuracy metric $h_\theta \leq \zeta(\alpha, H)$ and define the generalized exponential time scales as

$$
\theta_{\alpha, \nu} := \{\nu \alpha^k : k \in \mathbb{Z}\}
$$

where $\nu > 0$. Then for arbitrary $\mathcal{T}$ and $\forall \nu$, we have $\theta_{\alpha, \nu} \in \Gamma(\alpha)$ and

$$
\mathcal{A}_\mathcal{T}(\theta_{\alpha, \nu}) \leq 1 + \min_{\theta \in \Gamma(\alpha)} \mathcal{A}_\mathcal{T}(\theta).
$$

Figure 2.2 : (a) $\zeta^*(\alpha)$ vs. $\alpha$. For a large range of $\alpha$, $\zeta^*(\alpha)$ is close to 1. (b) Comparison of $L[\theta_{\alpha}](b)$ to its upper bound $C(b)$, for different values of $\alpha$. Observe that the lower bound of $L[\theta_{\alpha}](b)$ is almost identical to the upper bound $C(b)$. 

![Figure 2.2](image-url)
Moreover there exists $\xi > 0$ such that

$$A_T(\theta, \xi) = \min_{\theta \in \Gamma(\alpha)} A_T(\theta).$$  \hspace{1cm} (2.41)

The proof is in Appendix A.

Theorem 3 is a direct consequence of the fact that the accuracy metric $h_\theta$ increases with the largest ratio of consecutive time scales $d_\theta$. Thus $h_\theta = \zeta(d_\theta, H) \leq \zeta(\alpha, H)$ if and only if $d_\theta \leq \alpha$. Since the ratio of all consecutive time scale elements in $\theta_\alpha$ equals the maximum allowed value of $\alpha$, $\theta_\alpha$ is the most sparse among all sets $\theta$ with accuracy metric less than $\zeta(\alpha, H)$.

### 2.5 Asymptotics for fBm queues

In this section, for a queue with fBm input, we study the accuracy of the max, product, and sum approximations of $\mathbb{P} \{Q_\infty > b\}$ for asymptotically large queue thresholds, that is as $b \to \infty$. While asymptotic queuing results are not always directly applicable to scenarios with finite queues, they often provide intuition for network design [30, 44, 56, 60, 63].

We begin with some terminology. If $\lim_{b \to \infty} \Omega(b)/\Upsilon(b) = 1$ we say that $\Omega$ and $\Upsilon$ have the same asymptotic decay and denote it by $\Omega(b) \simeq \Upsilon(b)$. If $\log \Omega(b) \simeq \log \Upsilon(b)$ we say that $\Omega$ has the same log-asymptotic decay as $\Upsilon$. Under the assumption that $\Omega(b) \to 0$ it is easily shown that an asymptotic decay implies a log-asymptotic decay, that is,

$$\Omega(b) \simeq \Upsilon(b) \iff \frac{\Omega(b)}{\Upsilon(b)} \to 1 \implies \log \Omega(b) \simeq \log \Upsilon(b)$$  \hspace{1cm} (2.42)

but not vice versa. Note that “$\iff$” denotes equivalence. We call $\Upsilon$ an asymptotic upper bound of $\Omega$ if $\lim_{b \to \infty} \Omega(b)/\Upsilon(b) = 0$. 
2.5.1 Related work

Research on the asymptotic queuing behavior of fBm traffic has produced many enlightening results over the years. Large deviation principles reveal that $\mathbb{P}\{Q_{\infty} > b\}$ and $C(b)$ have the same log-asymptotic decay (see [30, 63])

$$\log \mathbb{P}\{Q_{\infty} > b\} \simeq \log C(b) \simeq -\frac{\eta b^{-(2-2H)}}{2}$$

(2.43)

where $\eta > 0$ is a constant depending on the traffic parameters and independent of $b$. However $\mathbb{P}\{Q_{\infty} > b\}$ and $C(b)$ do not have the same asymptotic decay: $\mathbb{P}\{Q_{\infty} > b\}$ is an asymptotic upper bound of $C(b)$. Interestingly under transient conditions, that is for a fixed $\tau$, $\mathbb{P}\{Q_{\tau} > b\}$ has the same asymptotic decay as $\sup_{0 \leq t \leq \tau} \mathbb{P}\{K_{t} - ct > b\}$ [28].

Recent results show that for fBm $\mathbb{P}\{Q_{\infty} > b\}$ has a Weibull asymptotic decay [44, 56, 60]

$$\mathbb{P}\{Q_{\infty} > b\} \simeq \vartheta b^{(1-H)(1-2H)/H} e^{-\eta b^{-2H}/2},$$

(2.44)

where $\vartheta > 0$ is a constant independent of $b$. When $1/2 < H < 1$, which implies that fBm’s increment process is LRD, this Weibull decay is slower than the exponential decay for a queue fed with short-range dependent traffic, for example fBm with $H = 1/2$ [50].

From (2.44) we obtain that $e^{-\eta b^{-2H}/2}$ is an asymptotic upper bound of $\mathbb{P}\{Q_{\infty} > b\}$ when $1/2 < H < 1$, since

$$\lim_{b \to \infty} \frac{b^{(1-H)(1-2H)/H} e^{-\eta b^{-2H}/2}}{e^{-\eta b^{-2H}/2}} = 0.$$  

(2.45)

This asymptotic upper bound was derived as the maximum variance approximation in [20]. For a detailed discussion on large queue asymptotics of LRD traffic see Chs. 4 to 11 of [7] and the references therein.


2.5.2 Asymptotic decay of multiscale queuing approximations

We now compare the log-asymptotic and asymptotic decay rates of the max, the product, and the sum approximations with that of \( P \{ Q_\infty > b \} \). We only consider the case \( \theta = \theta_\alpha \). The next theorem summarizes our results.

**Theorem 4** For a queue with fBm input traffic with parameters \( \bar{c}, \sigma, \) and \( H \), define

\[
b_k := \alpha^k \bar{c}(1 - H)/H, \quad k \in \mathbb{Z},
\]

(2.46)

where \( \alpha > 1 \) is arbitrary. Then the max, product, and sum approximations have the same log-asymptotic decay as \( P \{ Q^{[\theta_\alpha]} > b_k \} \) and \( P \{ Q_\infty > b_k \} \): that is as \( b_k \to \infty \) we have

\[
\log L^{[\theta_\alpha]}(b_k) \approx \log P^{[\theta_\alpha]}(b_k) \approx \log S^{[\theta_\alpha]}(b_k) \\
\approx \log P \{ Q^{[\theta_\alpha]} > b_k \} \approx \log P \{ Q_\infty > b_k \}.
\]

(2.47)

Moreover the max, product, and sum approximations all have the same asymptotic decay as \( P \{ Q^{[\theta_\alpha]} > b_k \} \): that is as \( b_k \to \infty \) we have

\[
L^{[\theta_\alpha]}(b_k) \approx P^{[\theta_\alpha]}(b_k) \approx S^{[\theta_\alpha]}(b_k) \approx P \{ Q^{[\theta_\alpha]} > b_k \}.
\]

(2.48)

However

\[
\lim_{k \to \infty} \frac{P \{ Q^{[\theta_\alpha]} > b_k \}}{P \{ Q_\infty > b_k \}} = 0.
\]

(2.49)

The proof is in Appendix A.

Theorem 4 reveals the strengths and limitations of using traffic statistics only at exponential time scales \( \theta_\alpha \) to capture queuing behavior. Recall from (2.2) and (2.6) that \( Q^{[\theta_\alpha]} \) approximates the queue size \( Q \) using traffic only at time scales \( t \in \theta_\alpha \). From (2.47) we see that \( \theta_\alpha \) is dense enough in \( \mathbb{R}_+ \) to ensure that \( P \{ Q^{[\theta_\alpha]} > b_k \} \) and \( P \{ Q_\infty > b_k \} \) have the same log-asymptotic decays for a particular unbounded increasing sequence...
of queue sizes $b_k$. However, $\theta_\alpha$ is not dense enough to ensure that $\mathbb{P}\{Q^{[\theta_\alpha]} > b_k\}$ and $\mathbb{P}\{Q_\infty > b_k\}$ have the same asymptotic decay.

We also observe from (2.48) that the max, product, and sum approximations have the same asymptotic decay as $\mathbb{P}\{Q^{[\theta_\alpha]} > b_k\}$. As a result they have the same log-asymptotic decay but different asymptotic decay as $\mathbb{P}\{Q_\infty > b_k\}$.

We next present non-asymptotic results comparing the different queuing approximations to $\mathbb{P}\{Q^{[\theta]} > b\}$.

### 2.6 Bounds for the queuing approximations

The knowledge of whether or not a queuing approximation is an upper or lower bound of $\mathbb{P}\{Q > b\}$ aids different applications. For example if we provision the queue service rate such that the critical time scale approximation $C(b)$ equals $10^{-6}$, then we must expect the actual tail queue probability $\mathbb{P}\{Q > b\}$ to exceed $10^{-6}$ since $C(b)$ lower bounds $\mathbb{P}\{Q > b\}$ (see (2.5)). If $C(b)$ is an accurate approximation of $\mathbb{P}\{Q > b\}$ to an order of magnitude, as our simulations with fBm traffic in Section 2.8 affirm, then we would effectively be provisioning for $\mathbb{P}\{Q > b\} < 10^{-5}$. If we replace the lower bound $C(b)$ by an approximation that is an upper bound of $\mathbb{P}\{Q > b\}$, then $\mathbb{P}\{Q > b\}$ is guaranteed to be less than $10^{-6}$.

In this section we prove bounding results for the max, the product, and the sum approximations, which we compare to $\mathbb{P}\{Q^{[\theta]} > b\}$ rather than $\mathbb{P}\{Q > b\}$. Note from (2.10) that lower bounds of $\mathbb{P}\{Q^{[\theta]} > b\}$ are also lower bounds of $\mathbb{P}\{Q > b\}$. While the queuing approximations that are upper bounds of $\mathbb{P}\{Q^{[\theta]} > b\}$ are not necessarily upper bounds of $\mathbb{P}\{Q > b\}$, they approximate $\mathbb{P}\{Q > b\}$ well as we show in Section 2.8.
2.6.1 Bounds for general input traffic processes

We first state a general result that holds for a queue fed by any traffic random process and then present model-specific results.

Lemma 5 For a discrete or continuous-time queue of infinite size, with an arbitrary input traffic process and constant service rate

\[ L_\tau^{[\theta]}(b) \leq \mathbb{P}\{Q_\tau^{[\theta]} > b\} \leq S_\tau^{[\theta]}(b) \]

(2.50)

and

\[ L_\tau^{[\theta]}(b) \leq P_\tau^{[\theta]}(b) \leq S_\tau^{[\theta]}(b), \]

(2.51)

where \( \theta \) is any countable subset of \( \mathbb{R}_+ \).

The proof is in Appendix A.

From Lemma 5 we see that max and sum approximations are always lower and upper bounds respectively of both \( \mathbb{P}\{Q_\tau^{[\theta]} > b\} \) and the product approximation. In the rest of this section we compare the product approximation to \( \mathbb{P}\{Q_\tau^{[\theta]} > b\} \).

Our results establish that for queues fed with fBm, WIG, or MWM input traffic, the product approximation is also an upper bound of \( \mathbb{P}\{Q_\tau^{[\theta]} > b\} \), like the sum approximation. For these three models, from Lemma 5 we then have

\[ L_\tau^{[\theta]}(b) \leq \mathbb{P}\{Q_\tau^{[\theta]} > b\} \leq P_\tau^{[\theta]}(b) \leq S_\tau^{[\theta]}(b), \]

(2.52)

implying that the product approximation is a closer upper bound of \( \mathbb{P}\{Q_\tau^{[\theta]} > b\} \) than the sum approximation.\(^\dagger\) We note from (2.12) and (2.13) that the product approximation has the added advantage that it is guaranteed to be less than or equal to 1 unlike the sum approximation.

\(^\dagger\)We prove (2.52) for the WIG and MWM only for \( \theta = \theta_2 \).
2.6.2 Bounds for fBm traffic

For queues fed with traffic from a large class of Gaussian processes, including fBm, $P_{\tau}^{[\theta]}(b)$ is an upper bound of $\mathbb{P}\{Q_{\tau}^{[\theta]} > b\}$ as claimed in (2.52).

**Theorem 6** Consider a Gaussian traffic process $X_\tau$ as input to an infinite buffer queue with constant service rate (discrete or continuous-time). If $\text{cov}(K_\tau[t], K_\tau[r]) \geq 0$ for all $t, r \in \theta$ then

$$\mathbb{P}\{Q_{\tau}^{[\theta]} > b\} \leq P_{\tau}^{[\theta]}(b),$$

(2.53)

where $\theta$ is any countable subset of $\mathbb{R}_+$. The proof is in Appendix A.

Note that fBm satisfies the requirements of Theorem 6 since

$$\text{cov}\left(K^{(dB)}[t], K^{(dB)}[r]\right) = \frac{1}{2}(t^{2H} + r^{2H} - |t - r|^{2H}) \geq 0$$

(2.54)

for all $t, r \geq 0$ and $0 < H < 1$.

2.6.3 WIG and MWM traffic

Recall from Section 2.3.3 that the WIG and MWM are non-stationary traffic models. As a consequence $P_{\tau}^{[\theta_2]}(b)$ changes with time location $\tau$. We first compare $P_{\tau}^{[\theta_2]}(b)$ to $\mathbb{P}\{Q_{\tau}^{[\theta_2]} > b\}$ for $\tau = 2^n$, that is at the final time instant of the tree process, and then at all other time instants $\tau$. We denote the final time instant $2^n$ by “end”.

**Theorem 7** For the WIG and MWM with arbitrary model parameters

$$\mathbb{P}\{Q_{\text{end}}^{[\theta_2]} > b\} \leq P_{\text{end}}^{[\theta_2]}(b) \ \forall b > 0.$$

(2.55)

The proof is in Appendix B.
Theorem 7 states that \( P^{[\theta_2]}_{\tau}(b) \) is an upper bound of \( \mathbb{P}\left\{ Q^{[\theta_2]}_{\tau} > b \right\} \) at the final time instant for the WIG and the MWM for arbitrary model parameters. The only ingredient of the proof of Theorem 7 is the fact that the quantities \( K_{\text{end}}[2^j], j = 1, 2, \ldots, n \) that determine \( P^{[\theta_2]}_{\text{end}}(b) \) are nodes along the right edge of the tree and hence are related through independent innovations (see Fig. 2.1). Since this fact is true for arbitrary model parameters, so is (2.55).

Generalizing the proof of Theorem 7 so that (2.55) holds for all time instants \( \tau \) is not straightforward because the quantities \( K_{\tau}[2^j], j = 1, 2, \ldots, n \) are not always tree nodes for arbitrary \( \tau \) and are hence not related through independent innovations as the quantities \( K_{\text{end}}[2^j], j = 1, 2, \ldots, n \) are.

However, on imposing certain restrictions on the WIG model parameters we can extend (2.55) to all \( \tau \) as stated next.

**Theorem 8** For the WIG model, if

\[
\text{var}(V_{0,0}) \geq \text{var}(Z_{0,0}) \quad \text{and} \quad \text{var}(Z_{j,k}) \geq 2\text{var}(Z_{j+1,k}),
\]

for \( j = 0, 1, \ldots, n-2 \) then

\[
\mathbb{P}\left\{ Q^{[\theta_2]}_{\tau} > b \right\} \leq P^{[\theta_2]}_{\tau}(b) \leq P^{[\theta_2]}_{\text{end}}(b), \quad \forall \tau.
\]

As a consequence

\[
\frac{1}{2^n} \sum_{\tau=1}^{2^n} \mathbb{P}\left\{ Q^{[\theta_2]}_{\tau} > b \right\} \leq P^{[\theta_2]}_{\text{end}}(b).
\]

The proof is in Appendix B.

Theorem 8 reveals that for a large class of WIG model parameters satisfying (2.56), \( P^{[\theta_2]}_{\text{end}}(b) \) is an upper bound of the time average of \( \mathbb{P}\left\{ Q^{[\theta_2]}_{\tau} > b \right\} \). A WIG model of fGn with \( 1/2 < H < 1 \) belongs to this class since it has

\[
\text{var}(Z_{j,k}) = \sigma^2 \left( T2^{-j(j+1)} \right)^{2H} (4 - 2^{2H})
\]

(2.59)
and

\[ \text{var}(V_{j,k}) = \sigma^2 (T2^{-j})^{2H}. \]  

(2.60)

Earlier work on the queuing behavior of the WIG model of fGn proved that the time average of the tail queue probability \( \mathbb{P}\{Q_t > b\} \) has the same log-asymptotic behavior as that of fGn [53]. In contrast to this result which is proved for asymptotically large \( b \) and only for a WIG model of fGn, Theorem 8 holds for any fixed queue threshold \( b \) and for more general WIG models. Similarities between the proof of Theorem 8 and the analysis in [53] are described in Appendix B.

We demonstrate through simulations in Section 2.8 that \( P_{\text{end}}(b) \) approximates the time average of \( \mathbb{P}\{Q_t > b\} \) well for a large range of queue sizes \( b \) for both the WIG and the MWM.

### 2.7 Convergence of the Product Approximation

In this section we study the convergence of the product approximation, \( P^{[\theta]}(b) \). We only consider the case of exponential time scales, that is \( \theta = \theta_\alpha \).

Recall that \( P^{[\theta_\alpha]}(b) \) is defined through a product of probabilities

\[ \prod_{k \in \mathbb{Z}} \mathbb{P}\{K[\alpha^k] - c\alpha^k < b\} \]  

(see (2.12)). Since \( 0 \leq \mathbb{P}\{K[\alpha^k] - c\alpha^k < b\} \leq 1 \), the question arises as to whether or not this product converges to 0 thus making \( P^{[\theta_\alpha]}(b) \) take the useless value of 1. We prove for fBm, WIG, and MWM-fed queues that for all \( b > 0 \) the terms \( \mathbb{P}\{K[\alpha^k] - c\alpha^k < b\} \) approach 1 fast enough to ensure that the product is positive and \( P^{[\theta_\alpha]}(b) \) is strictly less than 1.

#### 2.7.1 Product approximation of fBm traffic

We have already implicitly proved that the product approximation is less than 1 for asymptotically large queue thresholds \( b \) for an fBm-fed queue in Theorem 4 of Section
2.5. The same result holds for any fixed $b$ as stated below.

**Theorem 9** For a queue with fBm traffic as input, for all $b > 0$, $\alpha > 1$, and $0 < H < 1$

$$P^{[\theta, \alpha]}(b) < 1. \quad (2.61)$$

The proof is in Appendix A.

2.7.2 Product approximation of WIG traffic

We first spell out some notation. For a tree of depth $n$ we denote $P_{\text{end}}^{[\theta_2]}(b)$ by $\mathcal{P}^{(n)}(b)$, and $K_{\text{end}}[t]$ by $\mathcal{K}^{(n)}[t]$. Then we have

$$\mathcal{P}^{(n)}(b) = 1 - \prod_{k=0}^{n} \mathbb{P}\left\{ \mathcal{K}^{(n)}[2^k] - \mathcal{Z}^{(n)}2^k < b \right\}. \quad (2.62)$$

Clearly there is a possibility of $\mathcal{P}^{(\infty)}(b)$ taking the trivial value of 1.

We now prove that $\mathcal{P}^{(\infty)}(b)$ is strictly less than 1 for a WIG model of fGn. The proof is straightforward. Without loss of generality assume that $T = 1$. From the fact that $\mathcal{K}^{(n)}[2^k]$ has the same distribution as $B_{2k-n}$ we have that

$$\mathcal{P}^{(n)}(b) = 1 - \prod_{k=0}^{n} \mathbb{P}\left\{ B_{2k-n} - \mathcal{Z}^{(n)}2^k < b \right\} \leq 1 - \prod_{k=-\infty}^{\infty} \mathbb{P}\left\{ B_{2k-n} - \mathcal{Z}^{(n)}2^k < b \right\} = P^{(B),[\theta_2]}(b). \quad (2.63)$$

Since (2.63) holds for all $n$ we have from Theorem 9 that

$$\mathcal{P}^{(\infty)}(b) \leq P^{(B),[\theta_2]}(b) < 1. \quad (2.64)$$
We have so far only considered the convergence of the product approximation at the edge of the tree. For a WIG model of fGn with \(1/2 < H < 1\) from (2.57) and (2.64) it follows that

\[
P^{[\theta_2]}(b) \leq P^{(\infty)}(b)
\]

(2.65)

irrespective of the tree depth \(n\). Thus at no location \(\tau\) does the product approximation converge to 1.

### 2.7.3 Product approximation of MWM traffic

For the MWM with arbitrary parameters \(P^{(\infty)}(b)\) takes a non-trivial value as stated in the next theorem. Convergence of the product approximation at locations other than the edge of the MWM tree is beyond the scope of this paper.

**Theorem 10** For the MWM, there always exists a tree depth \(N\) such that

\[
\begin{align*}
P^{(N)}(b) & \leq P^{(\infty)}(b) \\
& \leq 1 - (1 - P^{(N)}(b))(1 - 2^{-N})^2 \\
& < 1.
\end{align*}
\]

(2.66)

The proof, which uses multifractal analysis, is in Appendix B.

We now complement our theoretical discussion of the different queuing approximations with simulation experiments.

### 2.8 Simulations

In this section we demonstrate the accuracy of the max, product, and sum approximations of \(\mathbb{P}\{Q > b\}\) through simulations with fGn, WIG, and MWM synthetic traces as well as with video and measured Internet traces. We also demonstrate that the tails of
Figure 2.3 : Comparison of the max the product and the sum approximations to $\mathbb{P}\{Q > b\}$ for fGn traffic with parameters $H = 0.8$, link capacity 10Mbps and $\sigma = 8 \times 10^5$ bits. For different utilizations in (a) and (b), the product and sum approximations are close to $\mathbb{P}\{Q > b\}$ for a wide range of queue thresholds $b$. The max approximation is a lower bound of $\mathbb{P}\{Q > b\}$ and is accurate to an order of magnitude.

Figure 2.4 : Histograms of the bytes-per-time processes at time-scale 2ms for (a) wide-area traffic at the University of Auckland (trace AUCK) [62], (b) one realization of the WIG model, and (c) one realization of the MWM. Note the large probability mass over negative values for the WIG model.

multiscale marginals of traffic have a significant impact on queuing in certain scenarios by comparing the queuing behavior of the WIG and MWM models with that of measured Internet traffic. We restrict our attention to exponential time-scales with $\alpha = 2$ (that is $\theta = \theta_2$). All error bars in the plots correspond to 95% confidence intervals.
2.8.1 Comparison of queuing approximations for fGn traffic

In earlier sections we theoretically compared the max, product, and sum approximations to $\mathbb{P}\{Q > b\}$ for an fGn-fed queue. We now compare the different approximations of $\mathbb{P}\{Q > b\}$ through simulations with fGn traffic.

**Simulation setup:** The simulations use fGn traces with Hurst parameter $H = 0.8$ and standard deviation at the 1s time-scale $\sigma = 8 \times 10^5$ bits that are generated using the method described in [25]. We set $t' = 10^{-4}$s and $c = 10$Mbps and vary the mean rate of the traces to obtain different utilizations.

We estimate $\mathbb{P}\{Q_\infty > b\}$ for each simulation run as the fraction of time for which the queue size exceeds $b$. To eliminate transients we only make estimates using queue sizes during the second half of the simulation. The plots of tail queue probability correspond to the mean obtained from 300 simulation runs. Each run uses a trace of length $2^{19}$ data points corresponding to a 52s simulation time.

**Simulation results:** The simulation results for two different utilizations are depicted in Fig. 2.3. We obtain the various queuing approximations using (2.8), (2.12), and (2.13) by choosing $\theta = \{t', 2t', \ldots, 2^{20}t'\}$ which is equivalent to $\theta_2$ truncated to lie within a fixed range of time-scales. Observe that in all cases $L[\theta_2](b)$ is a lower bound of $\mathbb{P}\{Q > b\}$ as predicted by (2.9). We also see that $L[\theta_2](b)$ is within an order of magnitude of $\mathbb{P}\{Q > b\}$ for a wide range of values of $\mathbb{P}\{Q > b\}$ ($\in [10^{-6}, 1]$). We conclude that $C(b)$ lying between $L[\theta_2](b)$ and $\mathbb{P}\{Q > b\}$ (see (2.9)) is also within an order of magnitude of $\mathbb{P}\{Q > b\}$ for the same range of $\mathbb{P}\{Q > b\}$.

From Fig. 2.3 observe that the product and sum approximations are almost identical and accurately track $\mathbb{P}\{Q > b\}$ for a wide range of queue sizes $b$. Also observe that they are better approximations than the max approximation in general. However unlike the max approximation, which is a guaranteed lower bound of $\mathbb{P}\{Q > b\}$, these
two approximations do not bound $\mathbb{P}\{Q > b\}$ from above or from below and in fact intersect it at some point. Call the queue threshold at which the product approximation and $\mathbb{P}\{Q > b\}$ intersect $b'$. We observe that in all cases the product approximation is greater than $\mathbb{P}\{Q > b\}$ at $b = 0$ and for $b > b'$ is always less than $\mathbb{P}\{Q > b\}$. Thus for $b > b'$ the product approximation lies between the max approximation and $\mathbb{P}\{Q > b\}$ which guarantees that it is a better approximation than the max approximation. The sum approximation has a similar behavior.

### 2.8.2 Impact of multiscale marginals on queuing

The impact of different traffic statistics on queuing has been extensively studied. Several studies have debated the importance of LRD for queuing [35, 43, 61, 66, 80]. LRD is however only a function of the asymptotic second-order correlation structure of traffic (or equivalently the variance of traffic at multiple time scales).

In this section we move beyond second-order statistics and demonstrate the importance of the tails of traffic marginals at different time scales on queuing. We do so by comparing the queuing behavior of the WIG and MWM processes with video and Internet WAN traces through simulations. Recall from Section 2.3 that both the WIG and
the MWM can capture a wide range of second-order correlation structures. They however differ in their marginal characteristics: the WIG process is Gaussian whereas the MWM process is non-Gaussian. We interpret our results using the product approximation and the conclusions of earlier work which studied the influence of link utilization on queuing [34].

**Traces:** The two traces we use are AUCK, which contains the number of bytes per 2ms of recorded WAN traffic (mostly TCP packets) [62] and VIDEO, which consists of 15 video clips multiplexed with random starting points [79]. The finest time-scale in VIDEO corresponds to 2.77ms, 1/15 the duration of a single frame. The mean rates of AUCK and VIDEO are 1.456Mbps and 53.8Mbps, respectively. AUCK contains $1.8 \times 10^6$ data points and VIDEO $2^{18}$. The Hurst parameter of AUCK obtained from the variance-time plot using time-scales 512ms to 262.144s is $H = 0.86$. For VIDEO, we find $H = 0.84$ using time-scales 354ms to 90.76s. From Fig. 2.4 and Fig. 2.5 observe that AUCK has a strongly non-Gaussian marginal while VIDEO’s marginal resembles a Gaussian distribution.

**Simulation results:** We fit the WIG and MWM to the real data and then generated synthetic traces from the models. We then compared the queuing behavior of the synthesized WIG and MWM traces with that of the real data when they are input to a FIFO queue of infinite length. The plots of $P\{Q > b\}$ correspond to the mean obtained from 1000 simulation runs.

We first present results for high link utilizations (> 70%). Observe from Figs. 2.6(a) and (b), where we used the WAN traffic trace AUCK, that the real and synthetic traces exhibit asymptotic Weibullian tail queue probabilities, in agreement with the theoretical findings for LRD traffic (compare (2.44)). However, apart from this asymptotic match, the MWM is much closer to the queuing behavior of the real trace. The link capacity
we use is 2Mbps, resulting in a utilization of 72%.

In the experiments with VIDEO (see Figs. 2.6(c) and (d)), which is much closer to a Gaussian process than AUCK, we observe that both the WIG and MWM closely match the correct queuing behavior. This confirms the influence of marginals and also reassures us that the MWM is flexible enough to model Gaussian traffic. Gaussian-like traffic, which must be positive, necessarily has a mean at least comparable to its standard deviation. Since for a large mean to standard deviation ratio the lognormal and Gaussian distributions resemble each other closely (see Fig. 2.5), the approximately lognormal MWM is suitable for Gaussian traffic [77]. The link capacity we use is 69Mbps, which corresponds to a utilization of 77%.

In the case of lower link utilizations (< 50%) from Fig. 2.7 we see that the MWM outperforms the WIG for both AUCK and VIDEO traces to a greater extent than in the high utilization case.

For both the MWM and WIG we observe that the product approximation is close to $P\{Q > b\}$ (see Figs. 2.6 and 2.7). The max approximation is within an order of magnitude of $P\{Q > b\}$.

**Interpretation using the product approximation:** Accepting the product approximation $P[\theta_x](b)$ as a close approximation to the actual tail queue probabilities, a closer look at (2.12) unravels how the marginals affect queue sizes. For traffic with heavier tailed marginals, the terms $P\{K[2^i] < b + c2^i\}$ are smaller and the product approximation is larger. Since the approximately lognormal MWM marginals are more heavy tailed than the Gaussian WIG marginals, the MWM has a larger product approximation than the WIG.

In the case of VIDEO, which shows marginals much closer to Gaussian (see Fig. 2.5), both the WIG and MWM perform similarly in terms of capturing the tail queue
probability at a high utilization, while at a low utilization the MWM outperforms the WIG. This result is easily explained using the finding in [34] that fine time-scale statistics influence queuing more than coarse time-scale statistics at low utilizations. Since fine time-scale marginals of VIDEO are more non-Gaussian than coarse time-scale marginals, obviously the MWM performs better than the WIG at low utilizations.

2.9 Summary

We have developed a new approach to queuing analysis of network traffic that uses traffic statistics at a fixed finite set of time scales. The queuing analysis provides three approximations for the tail queue probability of an infinite buffer queue with constant service rate. Theoretical and simulation results strongly support their use for different applications.

We also proved that exponential time scales are optimal for fBm traffic with respect to a trade-off in accuracy vs. computational cost of the max approximation. Applications can thus obtain accurate approximations to the tail queue probability by employing traffic statistics only at a few sparse exponential time scales.

Our simulations demonstrated the importance of multiscale marginals on queuing. We observed that in non-Gaussian traffic scenarios the correlation structure (short and long term) does not characterize the queuing behavior well.
Figure 2.6: Queuing performance of real data traces and synthetic WIG and MWM traces at high utilization. In (b), we observe that the MWM synthesis matches the queuing behavior of the AUCK data closely, while in (a) the WIG synthesis is not as close. In (c) and (d), we observe that both the WIG and the MWM match the queuing behavior of VIDEO. We also observe that the product approximation \( P^{\theta_2}(b) \) is close to the empirical queuing behavior for both synthetic traffic loads (both WIG and MWM) and that it performs better than the max approximation, \( L^{\theta_2}(b) \).
Figure 2.7: Queuing performance of real data traces and synthetic WIG and MWM traces at low utilization. The MWM outperforms the WIG even more than at higher utilizations.
Chapter 3

Sampling Theory

In this chapter we design strategies to optimally sample certain multiscale stochastic processes. Our results benefit applications that estimate the global average of some physical process (over space or time) from measured point samples of the process.

We begin by describing various multiscale stochastic processes in Section 3.1. Section 3.2 describes the water-filling technique to obtain optimal solutions for independent innovations trees and in Section 3.3 we prove optimal and worst case solutions for covariance trees. Through numerical experiments in Section 3.4 we demonstrate that optimal solutions for multiscale processes can vary depending on their topology and correlation structure. We describe related work on optimal sampling in Section 3.5. We summarize our results in Section 3.6. The proofs and pseudo-code for water-filling can be found in the Appendices.

3.1 Multiscale stochastic processes

Trees occur naturally in many applications as an efficient data structure with simple dependence structure. Of particular interest are trees which arise from representing and analyzing stochastic processes and time series on different time scales. Examples include the trees of wavelet coefficients as well as the tree of scaling coefficients for the Haar wavelet [54, 77, 90]. In this section we describe various trees and related background material relevant to this work.
3.1.1 Terminology and notation

A tree is a special graph, i.e., a set of nodes together with a list of pairs of nodes which can be pictured as directed edges pointing from one node to another with the following special properties (see Fig. 3.1). (1) There is a unique node called the root to which no edge points to. (2) There is exactly one edge pointing to any node other than the root. The starting node of the edge is called parent, the end point is called child. (3) The tree is connected, meaning that it is possible to reach any node from the root by following edges.

These simple rules imply that there are no cycles in the tree, in particular, there is exactly one way to reach a node from the root. Otherwise, rule (2) would be violated. Consequently, unique addresses can be assigned to the nodes which also reflect the level of a node in the tree. The topmost node is the root whose address we denote by $\phi$. Given an arbitrary node $\gamma$, its child nodes are said to be one level lower in the tree and are addressed by $\gamma k$ ($k = 1, 2, \ldots, P_\gamma$), where $P_\gamma \geq 0$. Each node is thus a concatenation of the form $\phi k_1 k_2 \ldots k_j$, or $k_1 k_2 \ldots k_j$ for short, where $j$ is the node’s scale or depth in the tree. The largest scale of any node in the tree is called the depth of the tree.

Nodes with no child nodes are termed leaves or leaf nodes. As usual, we denote the number of elements of a set of leaf nodes $L$ by $|L|$. We define the operator $\uparrow$ such that $\gamma k \uparrow = \gamma$. Thus, the operator $\uparrow$ takes us one level higher in the tree to the parent of the current node. Nodes that can be reached by repeated $\uparrow$ operations are called ancestors of $\gamma$. We term $\gamma$ a descendant of all of its ancestors.

The set of nodes and edges formed by $\gamma$ and all its descendants is termed the tree of $\gamma$. Clearly, it satisfies all rules of a tree. Let $L_\gamma$ denote the subset of $L$ that belong to the tree of $\gamma$. Let $N_\gamma$ be the total number of leaves of the tree of $\gamma$.

To every node $\gamma$ we associate a single (univariate) random variable $V_\gamma$. The case
where each node is associated with a multivariate random variable is beyond the scope of this thesis. For the sake of brevity we often refer to $V_\gamma$ as simply “the node $V_\gamma$” rather than “the random variable associated with node $\gamma$.”

## 3.1.2 Covariance trees

Covariance trees are multiscale stochastic processes defined on the basis of the covariance between the leaf nodes which is purely a function of the distance between them. Examples of covariance trees are the Wavelet-domain Independent Gaussian model and the Multifractal Wavelet models proposed for network traffic [54, 77]. Precise definitions follow. We denote variance, covariance, and expectation by “var”, “cov”, and “$E$” respectively.

**Definition 11** The distance between two leaf nodes is the scale of their lowest common ancestor.

**Definition 12** A covariance tree is a multiscale stochastic process with two properties.

1. The covariance of any two leaf nodes depends only on their distance. In other words, if the leaves $\gamma'$ and $\gamma$ have distance $l$ then

$$\text{cov}(V_\gamma, V_{\gamma'}) =: c_l.$$  

(3.1)
(2) All leaf nodes are at the same scale \( N \) and the root is equally correlated with all leaves.

In this chapter we consider covariance trees of two classes: trees with positive correlation progression and trees with negative correlation progression.

**Definition 13** A covariance tree has a positive correlation progression if \( c_l > c_{l-1} > 0 \) for \( l = 1, \ldots, N - 1 \). A covariance tree has a negative correlation progression if \( c_l < c_{l-1} \) for \( l = 1, \ldots, N - 1 \).

Intuitively in trees with positive correlation progression leaf nodes “closer” to each other in the tree are more strongly correlated than leaf nodes “farther apart.”

Our results take on a special form for covariance trees that are also symmetric trees.

**Definition 14** A symmetric tree is a multiscale stochastic process in which \( P_\gamma \), the number of child nodes of \( V_\gamma \), is purely a function of the scale of \( \gamma \).

### 3.1.3 Independent innovations trees

Independent innovations trees are particular multiscale stochastic processes defined as follows.

**Definition 15** An independent innovations tree is a multiscale stochastic process in which each node \( V_\gamma \), excluding the root, is defined through

\[
V_\gamma := \varphi_\gamma V_{\gamma |} + W_\gamma. \tag{3.2}
\]

Here, \( \varphi_\gamma \) is a scalar and \( W_\gamma \) is a zero-mean random variable independent of \( V_{\gamma |} \) as well as of \( W_{\gamma'} \) for all \( \gamma' \neq \gamma \). The root node is independent of \( W_\gamma \) for all \( \gamma \). In addition \( \varphi_\gamma \neq 0 \), \( \text{var}(W_\gamma) > 0 \ \forall \gamma \) and \( \text{var}(V_\emptyset) > 0 \) which together guarantee that \( \text{var}(V_\gamma) > 0 \ \forall \gamma \) as well as the linear independence of any set of tree nodes.
The fact that each node is the sum of a scaled version of its parent and an independent random variable makes these trees amenable to analysis [21, 90]. We prove optimality results for independent innovations trees in Section 3.2. Our results take on a special form for scale-invariant trees defined below.

**Definition 16** A scale-invariant tree is an independent innovations tree which is symmetric and where $\varrho_\gamma$ and the distribution of $W_\gamma$ are purely functions of the scale of $\gamma$.

While independent innovations trees are not covariance trees in general, it is easy to see that scale-invariant trees are indeed covariance trees with positive correlation progression.

### 3.2 Optimal leaf sets for independent innovations trees

In this section we determine the optimal leaf sets of independent innovations trees to estimate the root. We first describe the concept of water-filling which we later use to prove optimality results. We also present an efficient numerical method to obtain the optimal solutions.

#### 3.2.1 Water-filling

Concave optimization is a key ingredient of our analysis.

**Definition 17** A real function $\psi$ defined on the set of integers $\{0, 1, \ldots, M\}$ is concave if

$$\psi(x + 1) - \psi(x) \geq \psi(x + 2) - \psi(x + 1), \text{ for } x = 0, \ldots, M - 2. \quad (3.3)$$

The optimization problem we are faced with can be cast as follows. Given integers
\( P \geq 2, M_k > 0 \) \((k = 1, \ldots, P)\) and \( n \leq \sum_{k=1}^{P} M_k \) consider the discrete space

\[
\Delta_n(M_1, \ldots, M_P) := \left\{ X = [x_k]_{k=1}^{P} : \sum_{k=1}^{P} x_k = n, x_k \in \{0, \ldots, M_k\}, k = 1, 2, \ldots, P \right\}.
\]  

(3.4)

Given non-decreasing, concave functions \( \psi_k (k = 1, \ldots, P) \) with domains \( \{0, \ldots, M_k\} \) we are interested in

\[ h(n) := \max \left\{ \sum_{k=1}^{P} \psi_k(x_k) : X \in \Delta_n(M_1, \ldots, M_P) \right\}. \]

(3.5)

In the context of optimal estimation on a tree, \( P \) will play the role of the number of children that a parent node has, \( M_k \) the total number of leaf node descendants of the \( k \)-th child, and \( \psi_k \) the reciprocal of the optimal LMMSE of estimating the parent node given \( x_k \) leaf nodes in the tree of its \( k \)-th child.

The following iterative procedure constitutes a greedy algorithm that solves the optimization problem (A.2). Form vectors \( G^{(n)} = [g_k^{(n)}]_{k=1}^{P}, n = 0, \ldots, \sum_k M_k \) as follows:

Step (i): Set \( g_k^{(0)} = 0 \) \( \forall k \).

Step (ii): Set

\[ g_k^{(n+1)} = \begin{cases} 
g_k^{(n)} + 1, & k = m \\
g_k^{(n)}, & k \neq m \end{cases} \]

(3.6)

where

\[ m \in \arg \max_k \left\{ \psi_k \left( g_k^{(n)} + 1 \right) - \psi_k \left( g_k^{(n)} \right) : g_k^{(n)} < M_k \right\}. \]

(3.7)

The procedure described in Steps (i) and (ii) is termed water-filling because it resembles the solution to the problem of filling buckets with water to maximize the sum of the heights of the water levels. Buckets are narrow at the bottom and monotonously widen towards the top. Initially all buckets are empty (compare Step (i)). At each step
we are allowed to pour one unit of water into any one bucket with the goal of maximizing the sum of water levels. Intuitively at any step we must pour the water into that bucket which will give the maximum increase in water level among all the buckets not yet full (compare Step (ii)). Variants of this water-filling procedure appear as solutions to different information theoretic and communication problems [24].

**Lemma 18** The function $h(n)$ is non-decreasing and concave. In addition, $h(n) = \sum_k \psi_k \left( g_k^{(n)} \right)$ where $g_k^{(n)}$ is defined through water-filling.

The proof is in the Appendix C.

When all functions $\psi_k$ in Lemma 18 are identical, the maximum of $\sum_{k=1}^{P} \psi_k(x_k)$ is achieved by choosing the $x_k$’s to be “near-equal”. The following Corollary states this rigorously.

**Corollary 19** If $\psi_k = \psi$ for all $k = 1, 2, \ldots, P$ with $\psi$ non-decreasing and concave, then

$$h(n) = \left( P - n + P \left\lfloor \frac{n}{P} \right\rfloor \right) \psi \left( \left\lfloor \frac{n}{P} \right\rfloor \right) + \left( n - P \left\lfloor \frac{n}{P} \right\rfloor \right) \psi \left( \left\lfloor \frac{n}{P} \right\rfloor + 1 \right).$$

The maximizing values of the $x_k$ are apparent from (3.8). In particular, if $n$ is a multiple of $P$ then this reduces to

$$h(n) = P \psi \left( \frac{n}{P} \right).$$

The proof is in Appendix C.

Corollary 19 is key to proving our results for scale-invariant trees.

### 3.2.2 Optimal leaf sets through recursive water-filling

Our goal is to determine the choice of $n$ leaf nodes that gives the best linear minimum mean-square error (LMMSE) of the root. The LMMSE of $V_\gamma$ given $L_\gamma$ is defined as

$$\mathcal{E}(V_\gamma|L_\gamma) := \min_{\alpha} \mathbb{E}(V_\gamma - \alpha^T L_\gamma)^2,$$
where, in an abuse of notation, $\alpha^T L_\gamma$ denotes a linear combination of the elements of $L_\gamma$ with coefficients $\alpha$. Crucial to our proofs is the fact that $[21, 90]$

$$\frac{1}{\mathcal{E}(V_\gamma | L_\gamma)} + \frac{P_\gamma - 1}{\text{var}(V_\gamma)} = \sum_{k=1}^{P_\gamma} \frac{1}{\mathcal{E}(V_\gamma | L_{\gamma k})}. \quad (3.11)$$

We reproduce a proof of (3.11) from [21] in Appendix C for completeness.

Denote the set consisting of all subsets of leaves of the tree of $\gamma$ of size $n$ by $\Lambda_\gamma(n)$. Motivated by (3.11) we introduce

$$\mu_\gamma(n) := \max_{L \in \Lambda_\gamma(n)} \mathcal{E}^{-1}(V_\gamma | L) \quad (3.12)$$

and define

$$\mathcal{L}_\gamma(n) := \{L \in \Lambda_\gamma(n) : \mathcal{E}^{-1}(V_\gamma | L) = \mu_\gamma(n)\}. \quad (3.13)$$

Restated, our goal is to determine one element of $\mathcal{L}_\emptyset(n)$.

To allow a recursive approach through scale we generalize (3.12) and (3.13) by defining

$$\mu_{\gamma, \gamma'}(n) := \max_{L \in \Lambda_{\gamma, \gamma'}(n)} \mathcal{E}^{-1}(V_\gamma | L). \quad (3.14)$$

and

$$\mathcal{L}_{\gamma, \gamma'}(n) := \{L \in \Lambda_{\gamma, \gamma'}(n) : \mathcal{E}^{-1}(V_\gamma | L) = \mu_{\gamma, \gamma'}(n)\}. \quad (3.15)$$

Of course, $\mathcal{L}_\gamma(n) = \mathcal{L}_{\gamma, \gamma}(n)$. For the recursion, we are mostly interested in $\mathcal{L}_{\gamma, \gamma_k}(n)$, i.e., the optimal estimation of a parent node from a sample of leaf nodes of one of its children. The following will be useful notation

$$X^* := \arg \max_{X \in \Delta_n(N_\gamma, \ldots, N_{\gamma P_\gamma})} \sum_{k=1}^{P_\gamma} \mu_{\gamma, \gamma_k}(x_k). \quad (3.16)$$

Using (3.11) we can decompose the problem of determining $L \in \mathcal{L}_\gamma(n)$ into smaller problems of determining $L^{(k)} \in \mathcal{L}_{\gamma, \gamma_k}(x^*_k)$ as stated in the next theorem.
Theorem 20  For an independent innovations tree with $X^* = [x^*_k]_{k=1}^{P_\gamma}$ defined as in (3.16), $\bigcup_{k=1}^{P_\gamma} L^{(k)} \in \mathcal{L}_\gamma(n)$. Moreover, $\mathcal{L}_{\gamma k}(n) = \mathcal{L}_{\gamma k,\gamma k}(n) = \mathcal{L}_{\gamma,\gamma k}(n)$. Also $\mu_{\gamma,\gamma k}(n)$ is a positive, non-decreasing, and concave function of $n$, $\forall k, \gamma$.

The proof is in the Appendix C.

Theorem 20 provides the following recursive technique to construct $L \in \mathcal{L}_\gamma(n)$.

Starting at $\gamma$ we move downward determining how many of the $n$ leaf nodes of $L \in \mathcal{L}_\gamma(n)$ lie in the trees of the different descendants of $\gamma$ till we reach the bottom. Assume for the moment that we are given the functions $\mu_{\gamma,\gamma k}(n)$ for all $\gamma$. We present an efficient scheme to compute $\mu_{\gamma,\gamma k}(n)$ in Appendix E.

Scale-Recursive Water-filling scheme $\gamma \rightarrow \gamma k$

Step (a): Split $n$ leaf nodes between the trees of $\gamma k$, $k = 1, 2, \ldots, P_\gamma$.

First determine how to split the $n$ leaf nodes between the trees of $\gamma k$ by maximizing $\sum_{k=1}^{P_\gamma} \mu_{\gamma,\gamma k}(x_k)$ over $X \in \Delta(n, N_{\gamma k}, \ldots, N_{\gamma P_\gamma})$ (see (3.16)). The split is given by $X^*$ which is easily obtained using the water-filling procedure for concave functions (defined in (3.6)) since $\mu_{\gamma,\gamma k}(n)$ is concave for all $k$. Determine $L^{(k)} \in \mathcal{L}_{\gamma k}(x^*_k)$ since $L = \bigcup_{k=1}^{P_\gamma} L^{(k)} \in \mathcal{L}_\gamma(n)$.

Step (b): Split $x^*_k$ nodes between the trees of child nodes of $\gamma k$.

To our good fortune $L^{(k)} \in \mathcal{L}_{\gamma,\gamma k}(x^*_k)$ if and only if $L^{(k)} \in \mathcal{L}_{\gamma k}(x^*_k)$. Thus repeat Step (a) with $\gamma = \gamma k$ and $n = x^*_k$ to construct $L^{(k)}$. Stop when we have reached the bottom of the tree.

Efficient Implementation

We present an efficient implementation of the scale-recursive water-filling algorithm in Appendix E. This implementation first computes $L \in \mathcal{L}_\gamma(n)$ for $n = 1$ and then inductively obtains the same for larger values of $n$. Given $L \in \mathcal{L}_\gamma(n)$ we obtain $L \in \mathcal{L}_\gamma(n+1)$ as follows. Note from Step (a) above that we use water-filling to determine
how to split the $n$ leaves at $\gamma$. We are now required to split $n + 1$ leaves at $\gamma$. We easily obtain this from the earlier split of $n$ leaves using (3.6). The water-filling technique maintains the split of $n$ leaf nodes at $\gamma$ while adding just one leaf node to the tree of one of the child nodes (say $\gamma k'$) of $\gamma$. We thus have to perform Step (b) only for $k = k'$.

In this way the new leaf node “percolates” down the tree till we find its location at the bottom of the tree. The pseudo-code for determining $L \in \mathcal{L}_\gamma(n)$ given $\text{var}(W_\gamma)$ for all $\gamma$ as well as the proof that the recursive water-filling algorithm is polynomial-time is in Appendix E.

### 3.2.3 Uniform leaf nodes are optimal for scale-invariant trees

The symmetry in scale-invariant trees forces the optimal solution to take a particular form irrespective of the variances of the innovations $W_\gamma$. We use the following notion of uniform split to prove that in a scale-invariant tree a more or less equal spread of sample leaf nodes gives the best linear estimate of the root.

**Definition 21**

Given a scale-invariant tree, a vector of leaf nodes $L$ has uniform split of size $n$ at node $\gamma$ if $|L_\gamma| = n$ and $|L_{\gamma k}|$ is either $\lfloor \frac{n}{P_\gamma} \rfloor$ or $\lfloor \frac{n}{P_\gamma} \rfloor + 1$ for all values of $k$.

It follows that $\# \{ k : |L_{\gamma k}| = \lfloor \frac{n}{P_\gamma} \rfloor + 1 \} = n - P_\gamma \lfloor \frac{n}{P_\gamma} \rfloor$.

**Definition 22**

Given a scale-invariant tree, a vector of leaf nodes is called a uniform leaf sample if it has a uniform split at all tree nodes.

The next theorem gives the optimal leaf node set for scale-invariant trees.

**Theorem 23**

Given a scale-invariant tree, the uniform leaf sample of size $n$ gives the best LMMSE estimate of the tree-root among all possible choices of $n$ leaf nodes.

**Proof:** For a scale-invariant tree, $\mu_{\gamma_k}(n)$ is identical for all $k$ given any location $\gamma$. Corollary 19 and Theorem 20 then prove the theorem. $\square$
3.3 Covariance trees

In this section we prove optimal and worst case solutions for covariance trees. For the optimal solutions we leverage our results for independent innovations trees and for the worst case solutions we employ eigenanalysis. We begin by formulating the problem.

3.3.1 Problem formulation

Let us compute the LMMSE error of estimating the root $V_\phi$ given a set of leaf nodes $L$ of size $n$. Because of our assumption of a covariance tree, the correlation between any leaf node and the root node is identical. We denote this by $\rho$. Denote an $l \times j$ matrix with all elements equal to 1 by $1_{l \times j}$. It is well known [82] that the optimal linear estimate of $V_\phi$ given $L$ (assuming zero-mean random variables) is given by $ho 1_{1 \times n} S_L^{-1} L$, where $S_L$ is the covariance matrix of $L$ and that the resulting LMMSE is

$$E(V_\phi | L) = \text{var}(V_\phi) - \text{cov}(L, V_\phi)^T S_L^{-1} \text{cov}(L, V_\phi)$$

$$= \text{var}(V_\phi) - \rho^2 1_{1 \times n} S_L^{-1} 1_{n \times 1}. \quad (3.17)$$

Clearly obtaining the optimal and worst case choice for $L$ is equivalent to maximizing and minimizing the sum of the elements of $S_L^{-1}$. The exact value of $\rho$ does not affect the solution. We assume that no element of $L$ can be expressed as a linear combination of the others which implies that $S_L$ is invertible.

3.3.2 Optimal solutions

We use our results of Section 3.2 for independent innovations trees to determine the optimal solutions for covariance trees. Note from (3.17) that the estimation error for a covariance tree is a function only of the covariance between leaf nodes. Exploiting this fact, we first construct an independent innovations tree whose leaf nodes have the same
correlation structure as that of the covariance tree and then prove that both trees must have the same optimal solution. Previous results then provide the optimal solution for the independent innovations tree which is also the optimal for the covariance tree.

**Definition 24** A matched innovations tree of a given covariance tree with positive correlation progression is an independent innovations tree with the following properties. It has (1) the same topology (2) and the same correlation structure between leaf nodes as the covariance tree, and (3) the root is equally correlated with all leaf nodes (though the exact value of the correlation between the root and a leaf node may differ from that of the covariance tree).

All covariance trees with positive correlation progression have corresponding matched innovations trees. We construct a matched innovations tree for a given covariance tree as follows. Consider an independent innovations tree with the same topology as the covariance tree. Set \( \varrho_\gamma = 1 \) for all \( \gamma \),

\[
\operatorname{var}(V_{i}) = c_{0},
\]

and

\[
\operatorname{var}(W^{(j)}) = c_{j} - c_{j-1}, \; j = 1, 2, \ldots, N,
\]

where \( c_{j} \) is the covariance of leaf nodes of the covariance tree with distance \( j \) and \( \operatorname{var}(W^{(j)}) \) is the common variance of all innovations of the independent innovations tree at scale \( j \). Call \( c'_{j} \) the covariance of leaf nodes with distance \( j \) in the independent innovations tree. From (3.2) we have

\[
c'_{j} = \operatorname{var}(V_{i}) + \sum_{k=1}^{j} \operatorname{var}(W^{(k)}), \; j = 1, \ldots, N.
\]

We thus obtain \( c'_{j} = c_{j} \) for all \( j \) which proves that this independent innovations tree is the required matched innovations tree.
The next lemma relates the optimal solutions of a covariance tree and its matched innovations tree.

**Lemma 25** A covariance tree with positive correlation progression and its matched innovations tree have the same optimal leaf sets.

**Proof:** Note that (3.17) applies to any tree whose root is equally correlated with all its leaves. This includes both the covariance tree and its matched innovations tree. From (3.17) we see that the choice of $L$ that maximizes the sum of elements of $S^{-1}_L$ is optimal. Since $S^{-1}_L$ is identical for both the covariance tree and its matched innovations tree for any choice of $L$, they must have the same optimal solution. \qed

For a symmetric covariance tree that has positive correlation progression, the optimal solution takes on a specific form irrespective of the actual covariance between leaf nodes.

**Theorem 26** Given a symmetric covariance tree that has positive correlation progression, the uniform leaf sample of size $n$ gives the best LMMSE of the tree-root among all possible choices of $n$ leaf nodes.

**Proof:** Form a matched innovations tree using the procedure outlined previously. This tree is by construction a scale-invariant tree. The result then follows from Theorem 23 and Lemma 25. \qed

While the uniform leaf sample is the optimal solution for a symmetric covariance tree with positive correlation progression, it is surprisingly the worst case solution for certain trees with a different correlation structure which we prove next.
3.3.3 Worst case solutions

The worst case solution is any choice of $L \in \Lambda_\phi(n)$ that maximizes $E(V_\phi|L)$. We now highlight the fact that the optimal and worst case solutions can change dramatically depending on the correlation structure of the tree. Of particular relevance to our discussion is the set of clustered leaf nodes defined as follows.

**Definition 27** The set consisting of all leaf nodes of the tree of $V_\gamma$ is called the set of clustered leaves of $\gamma$.

We provide the worst case solutions for covariance trees in which every node (with the exception of the leaves) has the same number of child nodes. The following theorem summarizes our results.

**Theorem 28** Consider a covariance tree of depth $N$ in which every node (excluding the leaves) has the same number of child nodes $\sigma$. Then for leaf sets of size $\sigma^p$, $p = 0, 1, \ldots, N$ the worst case solution when the tree has positive correlation progression is given by the sets of clustered leaves of $\gamma$, where $\gamma$ is at scale $N - p$. The worst case solution is given by the sets of uniform leaf nodes when the tree has negative correlation progression.

The proof is in Appendix D.

Theorem 28 gives us the intuition that “more correlated” leaf nodes give worse estimates of the root. In the case of covariance trees with positive correlation progression, clustered leaf nodes are strongly correlated when compared to uniform leaf nodes. The opposite is true in the negative correlation progression case. Essentially if leaf nodes are highly correlated then they contain more redundant information which leads to poor estimation of the root.
While we have proved the optimal solution for covariance trees with positive correlation progression we have not yet proved the same for those with negative correlation progression. Based on the intuition just gained we make the following conjecture.

**Conjecture 29** *For a covariance tree that is negatively correlated, the sets of clustered leaf nodes correspond to the optimal solution.*

Using numerical techniques we support this conjecture in the next section.

### 3.4 Numerical results

In previous sections we proved that uniform leaf nodes were optimal for scale-invariant trees. In this section, using the scale-recursive water-filling algorithm we evaluate the optimal leaf sets for independent innovations trees that are not scale-invariant. In addition we provide numerical support for Conjecture 29.

#### 3.4.1 Independent innovations trees: scale-recursive water-filling

In this section we find the optimal leaf nodes to estimate the tree root for different independent innovations trees using the water-filling algorithm described in Section 3.2. We consider trees with depth \( N = 3 \) and in which all nodes have at most two child nodes. The results demonstrate that the optimal leaf sets are a function of the correlation structure and topology of the multiscale trees.

In Fig. 3.2(a) we plot the optimal leaf node sets of different sizes for a scale-invariant tree. As expected the uniform leaf nodes sets are optimal.

We consider a symmetric tree in Fig. 3.2(b), that is a tree in which all nodes have the same number of children (excepting leaf nodes). All parameters are constant within each scale except for the variance of the innovations \( W_\gamma \) at scale 1. The variance of the
optimal leaf sets (leaf set size)

(a) Scale-invariant tree (b) Unbalanced variance (c) Missing leaves

Figure 3.2: Optimal leaf sets for three different independent innovations trees: (a) scale-invariant tree, (b) symmetric tree with unbalanced variance of innovations at scale 1, and (c) tree with missing leaves at the finest scale. Observe that the uniform leaf node sets are optimal in (a) as expected. In (b), however, the nodes on the left half of the tree are more preferable to those on the right. In (c) the solution is similar to (a) for optimal sets of size \( n = 5 \) or lower but changes for \( n = 6 \) due to the missing nodes.

innovation on the right side is five times larger than the variance of the innovation on the left. Observe that leaves on the left of the tree are now preferable to those on the right and hence dominate the optimal sets. Comparing this result to Fig. 3.2(a) we see that the optimal sets are dependent on the correlation structure of the tree.

In Fig. 3.2(c) we consider the same tree as in Fig. 3.2(a) with two leaf nodes missing. These two leaves do not belong to the optimal leaf sets of size \( n = 1 \) to \( n = 5 \) in Fig. 3.2(a) but are elements of the optimal sets for \( n = 6 \) to 8. As a result the optimal sets of size 1 to 5 in Fig. 3.2(c) are identical to those in Fig. 3.2(a) whereas that for \( n = 6 \) differs. This result demonstrates that the optimal sets depend on the topology of the tree.

Our results have important implications for practical applications because situations arise where we must model physical processes using trees with different correlation structures and topologies. For example, if the process to be measured is non-stationary
over space then the multiscale tree may be unbalanced as in Fig. 3.2(b). In sensor networks it may not be possible to place sensors in certain locations due to physical constraints. We would thus have to exclude certain leaf nodes in our analysis as in Fig. 3.2(c).

3.4.2 Covariance trees: optimal and worst cases

This section provides numerical support for Conjecture 29 that states that the clustered node sets are optimal for covariance trees with negative correlation progression. We employ the wavelet-domain independent Gaussian (WIG) tree which is a covariance tree in which each node has $\sigma = 2$ child nodes [54]. We verify our claim using a WIG model of depth $N = 6$ possessing an fractional Gaussian noise-like correlation structure corresponding to Hurst parameter $H = 0.8$ and $H = 0.3$. To be precise, we choose the WIG model parameters such that the variance of nodes at scale $j$ is $2^{-2jH} \cdot \text{constant}$ (see [54] for further details). Note that $H > 0.5$ corresponds to positive correlation progression while $H \leq 0.5$ corresponds to negative correlation progression.

Fig. 3.3 compares the LMMSE of the estimated root node (normalized by the variance of the root) of the uniform and clustered sampling patterns. Since an exhaustive search of all possible patterns is computationally expensive (for example there are over $10^{18}$ ways of choosing 32 leaf nodes from among 64) we instead compute the LMMSE for $10^4$ randomly selected patterns. Observe that the clustered pattern gives the smallest LMMSE for the tree with negative correlation progression in Fig. 3.3(a) supporting our Conjecture 29 while the uniform pattern gives the smallest LMMSE for the positively correlation progression one in Fig. 3.3(b) corroborating Theorem 26. As proved in Theorem 28, the clustered and uniform patterns give the worst LMMSE for the positive and negative correlation progression cases respectively.
Figure 3.3: Comparison of probing schemes for a WIG model with (a) negative correlation progression and (b) positive correlation progression. Observe that the clustered nodes are optimal in (a) while the uniform is optimal in (b). The uniform and the clustered sets give the worst performance in (a) and (b) respectively, which confirms our theoretical results.

Our results confirm that the optimal solution is strongly dependent on the correlation structure of the tree. In many applications such as Internet traffic sampling using probe packets, one expects a positive correlation progression. Thus schemes that send a stream of back-to-back probe packets will give poor estimates of the average traffic rate. However in exceptional cases where the traffic could possess negative correlation progression, we conjecture that the same schemes will be optimal!

3.5 Related work

Earlier work has studied the problem of designing optimal samples of size $n$ to linearly estimate the global average of a process. For a one dimensional process which is wide-sense stationary with positive and convex correlation it was shown that systematic sampling (uniform patterns with random starting points) is optimal within a class of unbiased estimators [40].
For a two dimensional process on an \( n_1 \times n_2 \) grid with positive and convex correlation it was shown that an optimal sampling scheme does not lie in the class of schemes that ensure equal inclusion probability of \( \frac{n}{n_1 n_2} \) for every point on the grid [12]. Here an “optimal scheme” refers to a sampling scheme that achieves a lower bound on the error variance. The requirement of equal inclusion probability guarantees an unbiased estimator. The optimal schemes within certain sub-classes of this larger “equal inclusion probability” class were obtained using systematic sampling. More recent analysis refines these results to show that optimal designs do exist in the equal inclusion probability class for certain values of \( n, n_1, \) and \( n_2 \) and are obtained by Latin square sampling [49, 81].

Our results differs from the above works in that we provide optimal solutions for the entire class of linear estimators and study a different set of random processes.

A related problem of optimally sampling a process to minimize the entropy of the posterior process has been studied in the context of computer experiment design [92]. Two conditions that are sufficient for a sample to be universally optimal are that its elements are pair-wise equidistant and that this distance equal the maximum average distance of all possible sampling designs.

Other work on sampling fractional Brownian motion to estimate its Hurst parameter demonstrated that geometric sampling is superior to uniform sampling [86].

Recent work compared different probing schemes for traffic estimation through numerical simulations [42]. It was shown that a scheme which used uniformly spaced probes outperformed other schemes that used clustered probes. These results are similar to our findings for independent innovation trees and covariance trees with positive correlation progression.
3.6 Summary

We have addressed the problem of obtaining optimal leaf sets to estimate the root of two multiscale stochastic processes: independent innovations trees and covariance trees. Our findings have a wide range of applications including the design of sensor networks and Internet inference schemes.

We have proved for an independent innovations tree that the optimal solution can be obtained in polynomial-time by the water-filling algorithm. Our results show that the optimal solutions can vary drastically depending on the correlation structure of the tree. For covariance trees with positive correlation progression as well as scale-invariant trees we obtained that uniformly spaced leaf nodes are optimal. However, uniform leaf nodes give the worst estimates for covariance trees with negative correlation progression. Numerical experiments support our conjecture that clustered nodes provide the optimal solution for covariance trees with negative correlation progression.
Chapter 4

Network Probing

In this chapter we present two probing tools, pathChirp and STAB, to infer available bandwidth information about network paths. Both tools can be obtained as freeware from the web.

We describe our model for a network path in Section 4.1. Section 4.2 and 4.3 describe the working of pathChirp and STAB respectively. We summarize our contributions to the field of network probing in Section 4.4.

4.1 Network model and terminology

We focus on a single sender – single receiver path of a communication network. We explicitly permit multiple queues; to this end we model a path as a series of store-and-forward nodes each with its own constant service rate, equipped with FIFO queues.

Available bandwidth

Denote the capacity of the output queue of router node $i$ as $c_i$, and the total traffic (other than probes) entering it between times $a$ and $b$ as $A_i[a,b]$. Define the path’s available bandwidth in time interval $[t - \tau, t]$ as

$$B[t - \tau, t] = \min_i \left( c_i - \frac{A_i[t - \tau + p_i, t + p_i]}{\tau} \right), \quad (4.1)$$

where $p_i$ is the minimum time a packet sent from the sender could take to reach router $i$. The delay $p_i$ includes the speed-of-light propagation delay and packet service times at intermediate queues.
In reality probe packets suffer queuing delays in addition to the minimum delay $p_i$. Thus probes transmitted during $[t - \tau, t]$ can arrive at router $i$ outside time interval $[t - \tau + p_i, t + p_i]$ and do not exactly measure $B[t - \tau, t]$. For large $\tau \gg RTT$, however, the effect of queuing delay becomes inconsequential.

**Sub-path available bandwidth**

We define the sub-path available bandwidth up to link $i$ as the minimum available bandwidth among the first $i$ links of the path. The sub-path available bandwidth up to link $i$ is a non-increasing function of $i$. It decreases at all locations of thin links and remains constant between two consecutive thin link locations. The last thin link is obviously the link with the least available bandwidth on the entire path, the tight link.

### 4.2 pathChirp

In this section we present our first probing tool, pathChirp. We describe its working and demonstrate its efficiency and accuracy through Internet experiments and by comparing it to other tools through simulations.

#### 4.2.1 Algorithm

PathChirp estimates the available bandwidth along a path by launching a number of packet chirps (numbered $m = 1, 2, \ldots$) from sender to receiver and then conducting a statistical analysis at the receiver.

First some notation for chirps (see Figure 1.5(a)). Consider chirp $m$ consisting of $N$ exponentially spaced packets, each of size $P$ bytes. Define the ratio of successive packet inter-spacing times within a chirp as the spread factor $\gamma$, the queuing delay of packet $k$ as $q_k^{(m)}$, the sender transmission time of packet $k$ as $t_k^{(m)}$, the inter-spacing time between
packets $k$ and $k + 1$ as $\Delta_k^{(m)}$, and the instantaneous chirp rate at packet $k$ as

$$R_k^{(m)} = P / \Delta_k^{(m)}.$$  

(4.2)

If the clocks at the sender and receiver are unsynchronized but stable, then the difference between receiver and sender time stamps is the queuing delay plus a constant. Since $\Delta_k^{(m)}$ and $R_k^{(m)}$ are the same for all chirps, we drop their superscripts in the subsequent discussion.

In a CBR fluid cross-traffic scenario, we have

$$q_k^{(m)} = 0, \quad \text{if } B\left[t_1^{(m)}, t_N^{(m)}\right] \geq R_k$$

$$q_k^{(m)} > q_{k-1}^{(m)}, \quad \text{otherwise}$$

(4.3)

which leads to a simple estimate: $\hat{B}\left[t_1^{(m)}, t_N^{(m)}\right] = R_{k^*}$, where $k^*$ is the packet at which the queuing delay begins increasing.

The assumption of CBR cross-traffic clearly oversimplifies reality. In particular, due to bursty traffic, queuing delays will typically not increase monotonically within a chirp, or any probing train for that matter. Figure 4.1 depicts the queuing delays of a typical chirp train. We refer to such a plot as a queuing delay signature. Typically a signature consists of excursions from the zero axis ($q_k^{(m)} > 0$ for several consecutive packets) caused by bursts of cross-traffic. The first few excursions end with the queuing delays returning to zero. This is because the chirp rate $R_k$ is less than the bottleneck link speed ($c_{\text{min}} := \min\{c_i\}$) on the path, which allows the queues to relax in the absence of cross-traffic. The last excursion usually ends with increasing queuing delays because $R_k > c_{\text{min}}$, which causes the chirp packets to fill up intermediate queues.

PathChirp uses the shape of the signature, to make an estimate $E_k^{(m)}$ of the per-packet available bandwidth $B\left[t_k^{(m)}, t_{k+1}^{(m)}\right]$. It then takes a weighted average of the $E_k^{(m)}$‘s corresponding to each chirp $m$ to obtain estimates $D^{(m)}$ of the per-chirp available bandwidth.
Finally it makes estimates $B[t - \tau, t]$ of the available bandwidth $B[t - \tau, t]$ by averaging the estimates $D(m)$ obtained in the time interval $[t - \tau, t]$.

**Excursion segmentation**

In order to accurately compute $E_k^{(m)}$, pathChirp segments each signature into regions belonging to excursions and regions not belonging to excursions.

Based on the principle of self-induced congestion, we assume that increasing queuing delays signify less available bandwidth than the instantaneous chirp rate at that moment while decreasing delays signify the opposite, that is,

$$E_k^{(m)} \geq R_k, \quad \text{if } q_k^{(m)} \geq q_{k+1}^{(m)} \quad (4.5)$$

$$E_k^{(m)} \leq R_k, \quad \text{otherwise.} \quad (4.6)$$

In a single-hop scenario, (4.5) is exactly true while (4.6) need not always be true. For example if packets $k$ and $k+1$ are spaced very far apart (say by 1 hour), then the fact that $q_k^{(m)} < q_{k+1}^{(m)}$ tells us little about $E_k^{(m)}$. This is because the packets $k$ and $k+1$ cannot possibly induce congestion in the network and rather only provide independent samples of the path queuing delay.
To make correct use of (4.6), we segment each signature into excursion regions and apply (4.6) only to these regions. The basic idea behind pathChirp’s excursion segmentation algorithm is quite simple. Intuitively if $q_k^{(m)}$ increases and remains larger than 0 for several consecutive packets, then it is likely that these packets are all part of the same busy period* at a congested queue along the path. In this case we expect $q_k^{(m)} < q_{k+1}^{(m)}$ to correspond to self-induced congestion, thus validating (4.6). We would thus like to find regions in the signature for which $q_k^{(m)} > 0$ for several consecutive packets.

In practice we do not necessarily know the clock offset between the end hosts running pathChirp. This combined with the machine added noise to the time stamps makes it infeasible to use $q_k^{(m)} > 0$ for excursion detection. PathChirp instead uses the relative queuing delay within a chirp to detect excursions. It also avoids using hard queuing delay thresholds, since the magnitude of queuing delay is heavily dependent on link speeds that vary from path to path. For example, from basic queuing theory a 10Mbps link loaded at 50% utilization by a Poisson traffic source (with constant packet size) will have a larger average queuing delay than a similarly utilized 100Mbps link fed with Poisson traffic.

The details of pathChirp’s excursion segmentation algorithm are as follows. The goal is to identify potential starting and ending packet numbers $i$ and $j$ respectively for an excursion. Every packet $i$ where $q_i^{(m)} < q_{i+1}^{(m)}$ is a potential starting point of an excursion. We define the end of the excursion $j$ as the first packet where

$$q(j) - q(i) < \frac{\max_{i \leq k \leq j} [q(k) - q(i)]}{F},$$

where $F$ is a parameter called the \textit{decrease factor}. At $j$ the queuing delay relative to $q(i)$ has decreased by a factor of $F$ from the maximum queuing delay increase after $i$

\* A \textit{busy period} is a time interval during which the queue is never idle.
and up to $j$. If $j - i > L$, that is the signature region is long enough, then all packets between $i$ and $j$ form an excursion.

The last excursion of a signature usually does not terminate; that is, there is some packet $l$ with $q_{l}^{(m)} < q_{l+1}^{(m)}$ such that there is no $j > l$ for which (4.7) holds (replacing $i$ by $l$ in (4.7)). This excursion is treated differently to the others while setting $E_{k}^{(m)}$ which we describe next.

**Computing the per-packet estimates $E_{k}^{(m)}$**

Now it only remains to compute the per-packet available bandwidth estimates $E_{k}^{(m)}$. Each chirp packet $k$ falls into one of the following three categories that decide $E_{k}^{(m)}$.

**Case (a):** If $k$ belongs to an excursion that terminates and $q_{k}^{(m)} \leq q_{k+1}^{(m)}$, then set

$$E_{k}^{(m)} = R_{k}. \quad (4.8)$$

This satisfies (4.6).

**Case (b):** If $k$ belongs to an excursion that does not terminate, then set

$$E_{k}^{(m)} = R_{l}, \quad \forall k > l, \quad (4.9)$$

where $l$ is the start of the excursion.

The reason that we do not use (4.8) for case (b) is that the chirp rate during this particular excursion can be much higher than $c_{\text{min}}$. Since the available bandwidth is always less than $c_{\text{min}}$, we must have $E_{k}^{(m)} < R_{k}$.

We note however that according to (4.5) we must have $E_{k}^{(m)} > R_{k} > R_l$ if $q_{k}^{(m)} > q_{k+1}^{(m)}$, $k > l$. Hence (4.9) leads to a conservative estimate of $E_{k}^{(m)}$ for such $k$.

**Case (c):** For all $k$ not belonging to the above cases we set $E_{k}^{(m)} = R_{l}$. This includes all those $k$ not belonging to excursions as well as those with decreasing queuing delay belonging to excursions. In case the last excursion of the signature does terminate, we choose $l = N - 1$. 
For the pseudo-code of the pathChirp algorithm see Appendix F. Since the pseudo-code uses delay information of only a single chirp, we drop superscript $(m)$ in all quantities.

4.2.2 Implementation details

PathChirp infers available bandwidth online using UDP chirp packet probes. PathChirp’s parameters are the probe packet size $P$, the spread factor $\gamma$, the decrease factor $F$, the busy period threshold $L$, and the time interval $\tau$ over which the $D_m$ instantaneous estimates are smoothed. The average probe load on the network and the range of instantaneous rates within each chirp are user specified options. PathChirp spaces the chirps apart in time to achieve the specified average probing rate. Each UDP packet carries a sender timestamp which the receiver uses along with its own local timestamp in the delay estimation process.

In pathChirp, probe packets travel one-way from sender to receiver, and the receiver performs the estimation. We prefer to not merely echo back information to the sender to avoid the problem of echo probe traffic interfering with the sender-to-receiver chirp probes. This can occur on links that are not full-duplex, for example those in shared LANs.

PathChirp addresses the practical problem of context switching. When a context switch takes place at a host receiving probe packets, the packets are temporarily buffered while the CPU handles other processes. This introduces delays between packets reaching the application layer just before the context switch and after it. In addition, the buffered packets rapidly reach the application layer after the context switch. These delays may be mistakenly construed as router queuing delays and thus corrupt pathChirp’s network inference. When the difference between two consecutive receive time stamps
is less than a threshold \( d \), we detect a context switch and discard the concerned chirp. We note that a \( d \) value of 30\( \mu \)s is lower than the transmission time of a 1000 byte packet on an OC-3 link (50\( \mu \)s). Thus one would expect packet arrival times at the receiver to exceed 50\( \mu \)s if the last link is of OC-3 or lower speed. Currently \( d \) is hardcoded in the program. In future \( d \) will be adaptively chosen to suit the machine in question.

We are currently studying other ways of circumventing time stamp corruption due to context switching. One of them is to use time stamps generated by NIC cards rather than application layer ones.

PathChirp discards all chirps with dropped packets.

### 4.2.3 Performance and parameter choice

In this section, we use simulations to better understand the role of the various pathChirp parameters. In the experiments, we use a single queue with capacity 10Mbps fed with Poisson packet arrivals. The cross-traffic packet sizes were randomly chosen to be 1500 or 40 bytes with equal probability. Internet traffic has been shown to have weak correlations over time lags less than 100ms [93] in spite of stronger correlations (or long-range dependence (LRD)) at time lags greater than 1s. Since the duration of a chirp is typically less than 100ms a Poisson cross-traffic model which does not possess LRD suffices.

We varied the packet size \( P \), spread factor \( \gamma \), decrease factor \( F \) and busy period threshold \( L \) while keeping \( \tau \) and the total probing load constant at 500kbps. Recall that we can maintain any average low probing rate by spacing the chirp trains far enough apart. Our choice for the performance metric is the mean squared error (MSE) of the estimate \( \rho[0, \tau] \) normalized by the second moment of the true \( B[0, \tau] \). All experiments report 90% confidence intervals.

**Probe packet size \( P \):** First we assess the impact of probe packet size \( P \) on estimation
performance. Obviously the number of bytes transmitted per chirp decreases with $P$. Thus by reducing $P$ we can send more chirps for the same average probing rate, giving us more estimates $D^{(m)}$ per time interval $\tau$. However from (4.2) we observe that for the same set of probing rates $R_k$, a small $P$ results in a proportionately small $\Delta_k$. Intuitively the cross-traffic arriving over a time interval $\Delta_k$ is more bursty for smaller $\Delta_k$. For instance when $\Delta_k \to 0$ the cross-traffic process is far from smooth and to the contrary is a binary process: we either have one packet arriving or none at all. Thus shorter chirps will exhibit more erratic signatures and give less accurate estimates.

Fig. 4.2 demonstrates the effect of the probe packet size $P$ on estimation performance. We set $\gamma = 1.2$ and vary the parameters $F$ and $L$ as well as the link utilization. Observe that in most cases larger $P$ values give better performance. In a few cases in Fig. 4.2(a) the MSE increases slightly with $P$.

The results show that pathChirp generally performs better with larger packet sizes. In Internet experiments we thus use $P \geq 1000$ bytes.

**Spread Factor** $\gamma$: The spread factor $\gamma$ controls the spectrum of probing rates in a chirp. A smaller $\gamma$ leads to a dense spectrum of rates $R_k$, potentially increasing the accuracy of estimates $D^{(m)}$. It also leads to a finer sampling of network delay, thus potentially improving pathChirp’s ability to identify excursions. However it also increases the number of packets per chirp and hence reduces the number of estimates $D^{(m)}$ per time interval $\tau$, possibly degrading the estimate $\rho[t - \tau, t]$.

Fig. 4.3 demonstrates the effect of the spread factor $\gamma$ on estimation performance. We observe that the MSE decreases (that is, improves) with decreasing $\gamma$. This experiment uses $P = 1300$ byte packets. Since $\gamma > 2$ can give errors as high as 100% even in CBR scenarios, we have excluded them in the experiments.

PathChirp uses $\gamma = 1.2$ by default.
Figure 4.2: Normalized mean squared error vs. probe packet size $P$ for two utilizations: (a) 30% and (b) 70%. In most cases the MSE decreases with increasing packet size. The experiment used $\gamma = 1.2$.

Figure 4.3: Normalized MSE vs. spread factor $\gamma$ for two utilizations: (a) 30% and (b) 70%. The MSE decreases with decreasing $\gamma$.

**Busy period threshold** $L$ **and decrease factor** $F$: The busy period threshold $L$ and decrease factor $F$ influence pathChirp’s excursion segmentation algorithm. Recall that the $E_k^{(m)}$ estimates corresponding to an excursion region are always less than what they
would be if the region was not marked as belonging to one (compare cases (a) and (c) in Section 4.2.1). Increasing $L$ or decreasing $F$ makes it harder for bumps in signatures to qualify as valid excursions thus leading to over-estimates of the available bandwidth. Conversely, decreasing $L$ or increasing $F$ will lead to under-estimation of available bandwidth. The optimal choice for the busy period threshold $L$ and decrease factor $F$ will depend on the cross-traffic statistics at queues on the path.

From Figs. 4.4 and 4.5 observe that for our single queue Poisson cross-traffic scenario small values of $L$ and large values of $F$ give better performance.

Internet experiments indicate that the optimum values of $L = 3$ and $F = 6$ obtained from the above experiments provide overly conservative estimates of available bandwidth. This could possibly be due to the noise present in real experiments that is absent in simulations. The pathChirp tool instead uses $L = 5$ and $F = 1.5$ as default.

**Multi-Hop scenarios** Real Internet paths almost always are multi-hop. Although we are unaware of any rigorous study of the number of congested queues on typical Internet
paths, we hypothesize that congestion largely occurs at the edge of the network close to the source or receiver. Thus data packets might likely encounter two congested queues, one on each end of their paths. One fact that supports the argument that congestion occurs at the edge is that backbone ISPs have reported very low packet loss and queuing delay on their networks [65]. While it is possible for paths to have no congested queues or possibly one, it is important for tools like pathChirp to be robust to the presence of at least two congested queues along the end-to-end path.

This section tests pathChirp in a two-hop scenario as depicted in Fig. 4.6. As before, competing cross-traffic packet arrivals are Poisson and the packet sizes are chosen at random to be 1500 or 40 bytes with equal probability. The parameters we use are $\gamma = 1.2$, $L = 5$, $F = 2$, $P = 1500$ and $\tau = 3s$.

Each experiment consists of two scenarios. In the first, we load both queues with cross-traffic such that one queue has less available bandwidth (the *tight* queue) than the other (the *slack* queue). The slack queue essentially adds noise to the chirp packet.

Figure 4.5: Normalized MSE vs. decrease factor $F$ for two utilizations: (a) 30% and (b) 70%. The error improves with increasing $F$. 
delays. In the second, we set the cross-traffic rate at the slack queue to zero. An error of comparable magnitude in the two scenarios implies that pathChirp is robust to the noise of the slack queue in the first case.

In the first experiment the cross-traffic rate at the first queue is 30Mbps and that at the second queue 5Mbps. This sets the available bandwidth at the first queue (the tight one) to 10Mbps and that at the second (the slack one) to 15Mbps. From Fig. 4.7(a) we observe that the MSE is practically indistinguishable between the cases where the slack queue has 5Mbps cross-traffic and no cross-traffic at all.

In the second experiment both queues are fed with 10Mbps cross-traffic which sets the available bandwidth at the first queue to 30Mbps and that at the second to 10Mbps. From Fig. 4.7(b) to our surprise we observe that the MSE is marginally smaller when the slack queue is loaded that when it is not.

The results show that pathChirp is robust in multi-hop scenarios.

4.2.4 Comparison with TOPP

This section compares pathChirp with TOPP [59] using simulations when both use the same probing bit rate and probe packet spacings.

TOPP

TOPP sends out several packet pairs well-separated in time [59]. Denote the set of unique packet-pair spacings at the sender arranged in decreasing order as $\delta_k, k =$
Figure 4.7: Performance in multi-hop experiments. The MSE in the case of both queues being loaded is comparable to that when only one is loaded implying that pathChirp is robust to multi-hop paths.

1, \ldots, N-1 and the corresponding average spacings at the receiver as \( \eta_k, k = 1, \ldots, N-1 \). Then under the assumption of proportional sharing (see [59] for details) of bandwidth at all queues on the path, the plot of \( \frac{\eta_k}{\delta_k} \) vs. \( \frac{P}{\delta_k} \) is piecewise linear with increasing slope. The very first linear segment equals 1 for \( \frac{P}{\delta_k} \in (0, B[{-\infty, \infty}]) \), implying that the first breakpoint gives the available bandwidth \( B[{-\infty, \infty}] \). In practice the measured values of \( \frac{\eta_k}{\delta_k} \) will be noisy, making a statistical estimation of available bandwidth necessary. We employ the regression-based statistical estimation described in [58].

To compare pathChirp with TOPP, we keep probing loads the same and compute the MSE of the available bandwidth estimates over time intervals of length \( \tau \) seconds, that
is, $B[n\tau, (n+1)\tau]$, $n = 0, 1, \ldots, \infty$. We obtain TOPP’s estimate of $B[n\tau, (n+1)\tau]$ using only the probes transmitted during $[n\tau, (n+1)\tau]$. PathChirp’s estimates $\rho[n\tau, (n+1)\tau]$ are obtained as described in Section 4.2.1.

For pathChirp we fix the spread factor $\gamma$ and separate the chirps in time to maintain the desired average probing rate. For TOPP we use packet-pairs with the same inter-spacing times as the chirp packets, that is $\delta_k = \Delta_k$. The separation times between consecutive packet-pairs are chosen as independent exponentially distributed random variables.

**Single-hop scenarios**

This experiment uses a single queue with link speed 20Mbps fed with Poisson cross-traffic. The probe rate is 1Mbps and the pathChirp parameters are set to $P = 1500$ bytes, $\gamma = 1.2$, $F = 5$, and $L = 3$.

Fig. 4.8 displays the MSE for experiments with two different utilizations. Observe that the pathChirp outperforms TOPP by about an order of magnitude.

**Multi-hop scenarios** We next compare pathChirp and TOPP in the multi-hop scenario depicted in Fig. 4.6 with average probing rate 500kbps. In the first experiment we set the Poisson cross-traffic rates so that the first queue has 10Mbps available while the second has 15Mbps available. The first queue is thus the tight one. In the second experiment the rates are set so the the first queue has 20Mbps available and the second has 10Mbps available, thus making the second queue the tight one.

From Fig. 4.9 observe that again pathChirp outperforms TOPP like in the single-hop scenarios.

Since pathChirp uses queueing delay correlation information present in signatures and not just the average delay increase between packet pairs, the above results are not surprising. A theoretical analysis supporting these empirical findings is part of our future
Figure 4.8: Comparison of pathChirp and TOPP in a single-hop scenario for two utilizations: (a) 30% and (b) 70%. Observe that pathChirp performs far better than TOPP.

4.2.5 Comparison with pathload

We now compare pathChirp with pathload (version pathload_1.0.2) [45] using a simple test bed at Rice University depicted in Fig. 4.10. The goal is to compare their efficiency in terms of number of bytes used to obtain available bandwidth estimates of equal accuracy.

PathChirp and pathload differ in their measurement methodology as well as their output quantities. Recall that pathChirp uses chirp packet trains; each chirp probes the
Figure 4.9: Comparison of pathChirp and TOPP in multi-hop scenarios. In (a) the first queue has less available bandwidth than the second while in (b) the second has the least available bandwidth. Observe that pathChirp performs far better than TOPP.

path at a wide range of bit-rates. Also, pathChirp provides a single estimate of available bandwidth per specified time interval $\tau$. In contrast pathload employs long constant bit-rate (CBR) packet trains and adaptively varies the rates of successive packet trains in an effort to converge to the available bandwidth rate. Pathload provides minimum and maximum bounds on the available bandwidth while taking a variable amount of time to make the estimate.

We perform two sets of experiments to compare the tools. To measure the efficiency of the tools, in each experiment we compute the average number of bytes over 25 runs
that each tool takes to provide estimates accurate to 10Mbps.

To obtain the bytes used by pathload, we set its bandwidth resolution parameter to 10Mbps and take the average number of bytes used to make 25 estimates.

To count the bytes used by pathChirp, we employ the following procedure. Denoting the start of the experiment as time $0$, we compute the estimate $\rho[0, \tau]$ for different values of $\tau$. We define $\tau^*$ as that value of $\tau$ for which the difference between the 90 and 10 percentiles of $\rho[0, \tau]$ (obtained from 25 experiments) is less than 10Mbps. We then compute the number of probing bytes that pathChirp sends in a time interval of length $\tau^*$.

In this experiment pathChirp used default parameter values: $\gamma = 1.2$, $P = 1000$ bytes, $F = 1.5$, and $L = 5$ packets.

In the first set of experiments, we set the available bandwidth to a constant value using *iperf* CBR UDP traffic [2] while in the second set of experiments we employ Poisson UDP traffic [6]. The *iperf* packet size is 1470 bytes while that of the Poisson traffic is 1000 bytes. The results in Tables 4.1 and 4.2 indicate that pathChirp needs less than 10% of the bytes that pathload uses. In addition to the average number of bytes the two tools use to achieve the desired accuracy, Tables 4.1 and 4.2 provide the 10%-90%
Table 4.1: Efficiency comparison of pathChirp and pathload with iperf CBR cross-traffic.

<table>
<thead>
<tr>
<th>Available Bandwidth</th>
<th>Efficiency</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pathChirp</td>
<td>pathload</td>
</tr>
<tr>
<td>30Mbps</td>
<td>0.41MB</td>
<td>4.3MB</td>
</tr>
<tr>
<td>50Mbps</td>
<td>0.32MB</td>
<td>5.5MB</td>
</tr>
<tr>
<td>70Mbps</td>
<td>0.26MB</td>
<td>9.9MB</td>
</tr>
</tbody>
</table>

values of pathChirp estimates and the average of pathload’s minimum and maximum bounds of available bandwidth. Observe that pathChirp’s estimates have a consistent negative bias, implying that its measurements are conservative.

These results demonstrate pathChirp’s utility, especially for applications requiring rapid estimates of the available bandwidth using only a light probing load.

4.2.6 Internet experiments

This section describes Internet experiments with pathChirp. The experiments use the Y topology depicted in Figure 4.11. PathChirp is employed over a path from the Stanford Linear Accelerator Center (SLAC) to Rice University. To provide some control on the estimated bandwidth, we introduce Poisson traffic along a path from either Caltech or StarLight (Chicago) to Rice. At the time of the experiments, the Caltech-Rice path consisted of 14 layer-3 hops (from traceroute), the SLAC-Rice path consisted of 12 hops, and 4 of the links were shared. The StarLight-Rice path consisted of 9 hops of which 3 were common to the SLAC-Rice path.
Table 4.2: Efficiency comparison of pathChirp and pathload with Poisson cross-traffic.

<table>
<thead>
<tr>
<th>Available Bandwidth</th>
<th>Efficiency</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pathChirp</td>
<td>pathload</td>
</tr>
<tr>
<td>30Mbps</td>
<td>0.35MB</td>
<td>3.9MB</td>
</tr>
<tr>
<td>50Mbps</td>
<td>0.75MB</td>
<td>5.6MB</td>
</tr>
<tr>
<td>70Mbps</td>
<td>0.6MB</td>
<td>8.6MB</td>
</tr>
</tbody>
</table>

For this experiment, the pathChirp parameters are set as follows: $P = 1000$ bytes, $\gamma = 1.2$, $L = 5$ and $F = 1.5$. We choose $\tau$ to correspond to 11 chirp transmissions. The Poisson traffic packets are of size 1000 bytes.

In the experiment we sent bursts of Poisson traffic at different rates to study pathChirp’s ability to track the resulting changes in available bandwidth. The success is demonstrated in Figs. 4.12(a) and 4.12(b). Observe that the estimates decrease in proportion to the rate of the Poisson traffic, with stronger dips corresponding to larger Poisson...
Figure 4.12: (a) Available bandwidth estimates when Poisson traffic originates at Caltech. (b) Available bandwidth estimates when Poisson traffic originates at StarLight (Chicago). Observe that the pathChirp estimates fall in proportion to the introduced Poisson traffic.

4.3 STAB

In this section we present our second probing tool, STAB. We describe STAB’s algorithm and related work on thin link localizations. In addition we demonstrate STAB’s ability to locate thin links through simulations and Internet experiments.

4.3.1 Algorithm

STAB employs packet tailgating chirps to locate thin links. A chirp with each probe packet replaced by two packets, a large one followed closely by a small one, is called a packet tailgating chirp (see Figure 1.5(b)). We set the IP time-to-live (TTL) header
field of the large packets to $i$ while maintaining a large TTL value for the small packets. Since each router along the path decrements a packet’s TTL field by one and discards the packet if it has a TTL value of zero, the large packets in the tailgating chirp vanish after link $i$ while the small packets proceed to the receiver.

Packet tailgating chirps provide a simple scheme for estimating the sub-path available bandwidth up to link $i$. For the moment assume that we have time stamps of the probe packets indicating when they arrive at link $i$. By replacing the receiver time stamps in the pathChirp’s algorithm for available bandwidth with these time stamps, we obtain the sub-path available bandwidth up to link $i$.

While in practice we cannot obtain time stamps of packet arrivals at link $i$ for arbitrary $i$, to our good fortune we can closely approximate them with the receiver time stamps of the small tailgating packets. The reason for this is as follows. Recall from the principle of self-induced congestion that probe packets face an increasing queuing delay from congestion only if the probing bit rate exceeds the available bandwidth. Also note that since the large packets vanish after link $i$ the chirp probing bit rate decreases drastically after link $i$. As a result the chirp consisting of only small packets has a bit rate too low to induce congestion, and much queuing delay, after link $i$. Thus the small packets go through to the receiver with their interspacing at link $i$ remaining more or less unperturbed.

STAB initially determines the number of links along the path by incrementing the TTL of successive probe packets starting from one. Packets with TTL less than the number of links are dropped at links along the path due to TTL expiration while others make it to the receiver. The smallest TTL of all packets to reach the destination thus gives the number of links.

STAB then sends out tailgating chirps and varies the TTL of the large packets in
successive chirps to estimate the sub-path available bandwidth up to link $i$ for different
values of $i$ using the pathChirp algorithm described above.

Finally STAB determines the probability that link $i$ is a thin link as the fraction of
time within a specified time window for which the sub-path available bandwidth up to
link $i - 1$ is greater than that up to link $i$ by a multiplicative factor $\alpha$. The last link with a
high probability of being a thin link is most likely the tight link of the entire end-to-end
path. We choose $\alpha = 1.2$ in our experiments.

### 4.3.2 Related work on thin link localization

A recent study [10] used a tool, \textit{BFind}, to locate the tight link of a path. BFind essen-
tially induces network congestion through continuous transmission of UDP traffic and
determines the location of the tight link from traceroute round-trip times. Another tool
\textit{TReno} [57] emulates TCP using UDP packets with limited TTL fields and router ICMP
echo responses in order to locate tight links. Both these tools have the advantage that
they do not require a program to run at a receiver host at the end of the path unlike
STAB. They however have the drawback of introducing excessively large probe traffic
loads on the network which can potentially disrupt existing network traffic. We thus do
not test these tools and compare them to STAB in our Internet experiments.

The tool \textit{pipechar} [4] provides estimates of raw transmission bandwidth and avail-
able bandwidth at each link on an end-to-end path. Pipechar’s algorithm has not yet
been published to the best of our knowledge. Vis-à-vis STAB it has the advantage of not
requiring receiver host cooperation to run, but has the disadvantages of requiring routers
to respond with ICMP packets when they receive packets with TTL decremented to zero,
and also of requiring superuser privileges at the sender host. We compare pipechar to
STAB in our Internet experiments.
4.3.3 Validation of STAB through simulations

We now demonstrate STAB’s ability to perform spatio-temporal available bandwidth estimation through ns-2 simulations [5]. Internet experiments follow next. The simulation topology is designed to ensure that the tight link location changes over time.

We use the double web-farm topology depicted in Fig. 4.13 for the simulations. Each web-farm consists of 420 clients downloading data from 40 servers over a bottleneck link of 20Mbps. All other links in the web-farm have 40Mbps full-duplex bandwidth. Each web session consists of a client downloading 250 pages from a server. By choosing the page size from a heavy-tailed Pareto distribution we ensure that the generated traffic has a bursty “fractal” nature that is ubiquitously present in Internet traffic [51]. The interarrival times between page downloads are exponentially distributed. The web-farm is based on a topology provided in the ns-2 [5] suite of applications.

We set the utilization on the bottleneck link of each web-farm by choosing the number of web sessions appropriately. From Fig. 4.13 we see that the STAB probes travel across both web-farm bottlenecks before reaching their destination. All sources including the STAB source use 1000 byte packets. The average probing load is set to 300kbps in all simulations. This particular choice is arbitrary.
Fig. 4.14(a) depicts the actual sub-path available bandwidth up to link \( m \) for different intermediate links \( m \) and their variation over time. In the first half of the simulation, that is up to time \( t = 200 \)s, only the first web-farm generates traffic. As a result link 2 is the tight link of the path and consequently the available bandwidth plot flattens out after link 2 at any time prior to \( t = 200 \)s. We observe that the available bandwidth of the end-to-end path is about 15Mbps in this half of the simulation.

In the second half of the simulation both web-farms generate traffic. Because the
second web-farm generates more traffic than the first, link 5 now becomes the tight link. Observe from Fig. 4.14(a) that the available bandwidth plot dips at link 5 after time $t = 200s$. We observe that path available bandwidth is about 5Mbps in this half of the simulation.

From Fig. 4.14(b) we see that STAB estimates the sub-path available bandwidth well. We compute the sub-path available bandwidth up to link $m$ at any time instant by averaging the estimate of available bandwidth from the past 20 chirps which have the TTL of large packets set to $m$. Observe that prior to time $t = 200s$ the estimates flatten after link 2 while after $t = 200$ the estimates dip at link 5 due to the traffic from the second web-farm. On comparing Fig. 4.14(a) and (b) we see that STAB’s underestimates the available bandwidth of the first link by a small amount. This is explained by the fact that subsequent links on a path do have a non-negligible, though minor, influence on chirps consisting of only small tailgating packets, which we neglected in our earlier discussions. Note that because STAB generates estimates of sub-path available bandwidth only after time $t = 100s$, we do not plot information prior to this time instant in Fig. 4.14.

Plots such as Fig. 4.14(b) can prove very useful in optimizing network performance. For example after time $t = 200s$, by choosing an alternate route that bypasses the tight link, link 5, the receiver can potentially download data from the sender over a path with three times the available bandwidth. If the alternate route in addition bypasses the first thin link, link 2, then it can potentially have an available bandwidth eight times that of the current path. In practice we can obtain alternate routes through multi-homing, by using overlay networks, or with the help of mirror sites.

Using the estimates of sub-path available bandwidth we compute the probability that different links on the path are thin links. We compute the probabilities at any time instant
Figure 4.15: Probability of different links being thin links at time instants (a) $t = 180s$ and (b) $t = 360s$. We compute the probabilities from the sub-path available bandwidth in Fig. 4.14(b). In (a) only link 2 has a high probability of being a thin link, while in (b) both links 2 and 5 have high probabilities of being thin links.

using estimates of the sub-path available bandwidth in the past 100s. Recall that a link $m$ qualifies as a thin link if it has less available bandwidth than all preceding links, and that the thin link farthest away from the source is the tight link of the entire path. Fig. 4.15(a) plots the probability of different links being thin links at time instant $t = 180s$, which belongs to the first half of the simulation. We see that link 2 is almost certainly a thin link while the other links have low probabilities of being thin links. This strongly suggests that link 2 is the last thin link of the path and hence the tight link. Fig. 4.15(b) plots the probability of different links being thin links at time instant $t = 360s$, which belongs to the second half of the simulation. Now both links 2 and 5 are almost certainly thin links unlike other links. Clearly at this time instant link 5 is most likely the path’s tight link.
4.3.4 STAB Internet experiment

We next demonstrate STAB’s ability to locate the thin links of Internet paths. We run STAB simultaneously on two paths, one from the University of Wisconsin at Madison (UWisc) to Rice University and the other from the University of Illinois at Urbana-Champaign (UIUC) to Rice (see Fig. 4.16). The two paths share eight common links. The results correspond to a 30 minute experiment that began at 9a.m. on Tuesday, May 25, 2004. STAB uses an average probing load of 300kbps in this experiment.

Figure 4.16: Thin link localization experiment topology. The paths share eight links.

In Fig. 4.17 we plot STAB’s estimates of the sub-path available bandwidth over time for both paths. We compute the sub-path available bandwidth up to link \( m \) at any time instant using the estimates of available bandwidth from the past 30 chirps which have large packet TTL set to \( m \). The plots reveal several interesting facts. Observe that estimates of the sub-path available bandwidth are almost always less that 100Mbps. This is to be expected since the very first links of both paths are 100Mbps Ethernet links. Next observe that the sub-path available bandwidth dips at link 13 and 14 in Figs. 4.17(a) and (b) respectively after which the plots flatten out. This strongly suggests that these links are the tight links of the two paths. In fact, both these correspond to the
Figure 4.17: STAB estimates of sub-path available bandwidth for the (a) UWisc – Rice and (b) UIUC – Rice paths corresponding to the topology depicted in Fig. 4.16. In (a) observe a steep drop at link 13 and in (b) at link 14 indicating that they are thin links.

same 100Mbps Fast Ethernet link within Rice University (see Fig. 4.16). Thus STAB’s estimates for the two paths are consistent.

We confirm the intuition gained from Fig. 4.17 about the location of the tight links of the two end-to-end paths through plots of the probability of different links being thin links. We compute the probabilities at any time instant using estimates of the sub-path available bandwidth in the past 3.5 minutes. From Figs. 4.18 and 4.19 we observe that at different time instants in the experiment, indeed link 13 and link 14 are the last links
Figure 4.18: Probability of different links being thin links for the UWisc – Rice path at time instants (a) $t = 10\text{min}$ and (b) $t = 20\text{min}$.

Figure 4.19: Probability of different links being thin links for the UIUC – Rice path at time instants (a) $t = 10\text{min}$ and (b) $t = 20\text{min}$.

with a high probability of being thin links for the UWisc – Rice and UIUC – Rice paths respectively. These links are located close to the edge of the end-to-end path thus supporting the intuition that congestion normally occurs at the edge of the network.

The link utilization data we obtain from routers supports our tight link inferences.
This data corresponds to the average utilization of links in 5 minute intervals and is obtained using the Multi Router Traffic Grapher (MRTG) [3] tool. We obtain MRTG data from all links in the Abilene, the Texas GigaPOP, and the Rice University networks belonging to the two paths except for two OC-12 layer-2 links within the Texas GigaPOP. These layer-2 links are two of four layer-2 links that comprise link 12, which is a layer-3 link, of the UWisc – Rice path in Fig. 4.16. Among all the links we have MRTG data from, link 13 of the UWisc – Rice path indeed has the least available bandwidth, about 80Mbps. We observe, however, that STAB underestimates its available bandwidth to be about 50Mbps (see Fig. 4.17). Understanding the causes of this underestimation is part of our ongoing work.

Finally we compare STAB to the tool pipechar. We ran pipechar twice to locate the tight link on the UWisc – Rice path immediately after the conclusion of our experiment with STAB. Pipechar estimates that link 12 has the least available bandwidth on the path, slightly less than the available bandwidth of link 13. In the two runs pipechar estimates the available bandwidth at link 12 to be 45.8Mbps and 59.4Mbps, and that of link 13 to be 59.4Mbps and 61.2Mbps. Pipechar’s estimates for the available bandwidth of link 13 corroborate STAB’s available bandwidth estimates for the same link. We cannot verify pipechar’s estimates for link 12 because of incomplete MRTG data as mentioned above.

### 4.4 Summary

We have presented pathChirp, an active probing scheme that uses a novel “packet chirp” strategy to dynamically estimate the available bandwidth along an end-to-end network path. Internet and testbed experiments as well as simulations reveal that pathChirp provides accurate, though somewhat conservative, estimates of the available bandwidth. In addition, pathChirp outperforms existing tools in terms of estimation accuracy and
efficiency.

We have also presented STAB a tool to locate thin links in space and over time. It has a strong potential to improve numerous network aware applications and assist in network trouble-shooting.
Chapter 5

Future Directions

This thesis has developed novel multiscale techniques to address various problems in queuing analysis, sampling theory, and network probing. Our work has opened up new avenues for future research in each of these areas. We highlight some of these in this section.

5.1 Queuing analysis

Our multiscale queuing analysis currently provides approximations to the tail queue probability only of a single queue. Extending the analysis to a network of queues is one challenging item for future work. To compute our queuing approximations we require the marginals of traffic at the input of the queue at a fixed set of time scales. In order to apply our approximations to a network of queues we must know the marginal distributions of traffic at the input of all queues in the network.

The problem is that the marginals of a traffic stream can get deformed as the stream goes through a queue. In case very little queuing occurs in the network, as is believed to be the case in the Internet backbone, we can use the same marginals for a particular traffic stream throughout the network [38, 91]. In case the effects of queuing are non-negligible, however, we must compute the marginals of each traffic stream at the output of a queue from the statistics of traffic at the input and the queue’s service rate. Finding a good solution to this problem will contribute significantly to the growing field of
stochastic network calculus [52, 83].

5.2 Sampling theory

Our work on optimal sampling strategies raises several interesting questions for future research. While the general problem of determining the $n$ best random variables to linearly estimate another random variable is an NP-hard problem, we devised a polynomial-time algorithm to solve one problem of this type, namely determining the optimal leaf set for an independent innovations tree. Clearly, the structure of independent innovations trees was an important factor that enabled a fast algorithm. The question arises as to whether there are similar problems that have polynomial-time solutions.

We have proved optimal results for covariance trees by reducing the problem to one for independent innovations trees. Such techniques of reducing one optimization problems to another problem that has an efficient solution can be very powerful. If a problem can be reduced to one of determining optimal leaf sets for independent innovations trees in polynomial-time, then its solution is also polynomial-time. Which other problems are malleable to this reduction is an open question.

Sensor networks face several challenging research problems. We have only partially solved the issues of sensor placement. Ideally this problem must be solved in conjunction with other important problems such as developing optimal communication strategies and protocols.

Inference of the Internet’s internal properties is a rapidly developing field. While cross-traffic inference fits into our paradigm of estimation on trees, other inference problems require a more general framework. For example, a network operator may want to know the optimal $n$ locations in his network to capture detailed traffic traces. Detailed traffic traces are, however, too complicated to be represented by a single random vari-
able on a tree as in our paradigm. An important task of future work is to generalize our theory to arbitrary statistical models and optimality criteria.

5.3 Network probing

The current algorithm of pathChirp for available bandwidth estimation mainly uses information about whether delays are increasing or decreasing in the chirp delay signatures. The tool can be enhanced by more fully exploiting the rich information contained in the signatures.

Our probing tools currently provide available bandwidth information for only a single network path. Combining pathChirp and STAB with network tomography [23] will provide detailed maps of the Internet. Network tomography transmits probes between multiple sender and receiver hosts to determine various internal properties of the network. This is akin to medical imaging tomography, where X-rays or some other form of radiation is sent through a patient from different angles and the results are combined to obtain a detailed internal 3D picture of the patient.

Adapting the tools for use in wireless networks is also an important direction to pursue. Our tools are based on the principle of self-induced congestion which assumes that network delays of packets are mainly caused by queue build-ups at routers. Although this may be true in wired networks, the assumption may not hold in wireless networks where poor channel quality and interference from neighboring wireless hosts can delay packets. Probing in wireless networks is a nascent research area with several unexplored and challenging problems.
Appendix A

Proofs of queues with Gaussian traffic as input

Proof of Theorem 1

We prove the theorem in four steps.

**Step I:** Determine $\inf_{t>0} g(b, t)$ for a fixed value $b > 0$.

From (3.4) we obtain the partial derivative of $g(b, t)$ with respect to $t$:

$$
\frac{\partial g(b, t)}{\partial t} = t^H \hat{c} - \left( b + \hat{c} t \right) H t^{H-1} \sigma t^{2H} = \frac{\hat{c} t (1 - H) - b H}{\sigma t^{1+H}} \begin{cases} 
< 0, & 0 < t < \lambda(b) \\
= 0, & t = \lambda(b) \\
> 0, & t > \lambda(b)
\end{cases}
$$

where

$$
\lambda(b) = \frac{b H}{\hat{c} (1 - H)}.
$$

Note that $\lambda(b) > 0$ because $\hat{c} > 0$ and $0 < H < 1$ by assumption. Thus for a fixed value of $b$, $g(b, t)$ is monotonically decreasing over $0 < t < \lambda(b)$ and monotonically increasing when $t > \lambda(b)$. Clearly

$$
\inf_{t>0} g(b, t) = g(b, \lambda(b)) = \frac{b + \hat{c} \frac{b H}{\sigma (1-H)}}{\sigma \left( \frac{b H}{\sigma (1-H)} \right)^H} = \frac{b(1 - H + H) \left( \frac{\hat{c} (1 - H)}{b H} \right)^H}{\sigma H^{H(1 - H)} (1 - H)^{1-H}}.
$$

Note that $\lambda(b)$ is indeed the critical time scale defined by (2.3).

**Step II:** Determine $\varsigma(b) := \inf_{t \in \Theta} g(b, t) / g(b, \lambda(b))$ for fixed $b$.

Observe from (2.33) that the sequence $\{t_k\}_{k \in \mathbb{Z}}$ extends from $0$ to $\infty$. Thus there must exist an $l \in \mathbb{Z}$ such that $\lambda(b) \in [t_{l-1}, t_l]$. 


Consider the function
\[ f(b, t) := \frac{g(b, t)}{g(b, \lambda(b))}. \] (A.4)

Keeping \( b \) fixed, from (A.1) we see that \( g(b, t) \) increases as we move away from \( t = \lambda(b) \), we must have that
\[ \varsigma(b) = \min \{ f(b, t_{l-1}), f(b, t_{l}) \}. \] (A.5)

**Step III:** Determine \( \sup_{b \in \mathcal{A}_l} \varsigma(b) \) where
\[ \mathcal{A}_l := [\lambda^{-1}(t_{l-1}), \lambda^{-1}(t_l)], \] (A.6)
and \( \lambda^{-1}(t) \) is the inverse of \( \lambda(b) \) given by
\[ \lambda^{-1}(t) := \tilde{c} t (1 - H)/H. \] (A.7)

Note that \( b \in \mathcal{A}_l \) is equivalent to \( \lambda(b) \in [t_{l-1}, t_l] \). Since (A.5) holds for all \( b \in \mathcal{A}_l \) we see that \( f(b, t_{l-1}) \) and \( f(b, t_{l}) \) decide \( \sup_{b \in \mathcal{A}_l} \varsigma(b) \). From (3.4), (A.3) and (A.4) we have
\[ f(b, t) = \frac{b + \tilde{c} t}{b^{1-H}} \cdot \mathcal{R} \] (A.8)
where \( \mathcal{R} \) is a positive expression not depending on \( b \). Fixing \( t \) and differentiating with respect to \( b \) we get
\[
\frac{1}{\mathcal{R}} \cdot \frac{\partial f(b, t)}{\partial b} = \frac{b^{1-H} - (b + \tilde{c} t)(1 - H)b^{-H}}{b^{2-H}} = \frac{b^{-H}(b - (b + \tilde{c} t)(1 - H))}{b^{2-H}} = \frac{bH - \tilde{c} t (1 - H)}{b^{2-H}} \left\{ \begin{array}{ll} < 0, & 0 < b < \lambda^{-1}(t) \\ = 0, & b = \lambda^{-1}(t) \\ > 0, & b > \lambda^{-1}(t). \end{array} \right. \] (A.9)

Clearly for fixed \( t \), \( f(b, t) \) is monotonically decreasing with \( b \) for \( 0 < b < \lambda^{-1}(t) \) and monotonically increasing for \( b > \lambda^{-1}(t) \).
Using (A.9) we now compute \( \sup_{b \in A_l} \zeta(b) \). From (A.6) and (A.9) observe that \( f(b, t_{l-1}) \) monotonically increases and \( f(b, t_l) \) monotonically decreases as \( b \) increases over \( A_l \). If there exists \( a_l \in A_l \) such that

\[
 f(a_l, t_{l-1}) = f(a_l, t_l),
\]

then \( \zeta(b) \) must attain its supremum over \( A_l \) at this point (from (A.5)). Indeed such an \( a_l \) does exist. For the ease of notation we use

\[
 s_l := t_l / t_{l-1}.
\]

Solving (A.10), that is,

\[
 \frac{a_l + \tilde{c}t_{l-1}}{\sigma t_{l-1}^H} = \frac{a_l + \tilde{c}t_l}{\sigma t_l^H}
\]

we obtain \( a_l \) as

\[
 a_l = \frac{\tilde{c}t_{l-1} t_l (t_l^{-1} - t_l^{-H-1})}{t_l^H - t_{l-1}^H} = \tilde{c}t_l \cdot \frac{1 - s_l^{H-1}}{s_l^H - 1} = \frac{\tilde{c}t_l}{s_l} \cdot \frac{s_l - s_l^H}{s_l^H - 1}. \tag{A.13}
\]

As a result

\[
 g(a_l, t_l) = \frac{a_l + \tilde{c}t_l}{\sigma t_l^H} = \frac{\tilde{c}t_l \left( \frac{1 - s_l^{H-1}}{s_l^H - 1} \right) + 1}{\sigma t_l^H} = \frac{\tilde{c}t_l^{1-H} (s_l^H - s_l^{H-1})}{\sigma (s_l^H - 1)} = \frac{\tilde{c}t_l^{1-H} s_l^{H-1} (s_l - 1)}{\sigma (s_l^H - 1)}. \tag{A.14}
\]

Then from (2.35), (A.3), (A.13), and (A.14),

\[
 \sup_{b \in A_l} \zeta(b) = f(a_l, t_l) = \frac{g(a_l, t_l)}{g(a_l, \lambda(a_l))} = \left( \frac{s_l - 1}{s_l^H - 1} \right) \frac{\tilde{c}t_l^{1-H}}{\sigma s_l^{1-H}} \cdot \frac{\sigma H^H (1 - H)^{1-H}}{\tilde{c}H \left( \frac{\tilde{c}t_l}{s_l} \right)^{1-H} \left( \frac{s_l - s_l^H}{s_l^H - 1} \right)^{1-H}} \tag{A.15}
\]

\[
 = \frac{s_l - 1}{(s_l^H - 1)^H} \cdot \frac{1}{(s_l - s_l^H)^{1-H}} \cdot H^H (1 - H)^{1-H} = \zeta(s_l, H)
\]

**Step IV:** Determine \( h_\theta = \sup_{b \in \mathbb{R}_+} \zeta(b) \).
Claim 30 $\zeta(s_l, H)$ increases with $s_l$.

Proof of Claim 30: Note from (A.15) that $\zeta(s_l, H)$ equals $f(a_l, t_l)$. It is thus sufficient to prove that $f(a_l, t_l)$ increases with $s_l$. Without loss of generality we study how $f(a_l, t_l)$ changes by varying $t_{l-1}$ keeping $t_l$ fixed. Note that this is equivalent to varying $s_l$. We have from (A.13)

$$\frac{1}{c_{t_l}} \frac{\partial a_l}{\partial t_{l-1}} = \frac{(t_l^H - t_{l-1}^H)(H_{l-1}^H - t_{l-1}^H) - (t_{l-1}^H - t_{l-1} t_l^H)(-H t_{l-1}^H)}{(t_l^H - t_{l-1}^H)^2} = \frac{H t_{l-1}^H t_l^H}{(t_l^H - t_{l-1}^H)^2} - t_l^H t_{l-1}^H - H t_{l-1}^H t_l^H + H t_{l-1}^H t_l^H H t_{l-1}^H - H t_{l-1}^H t_l^H$$

It is easily shown that the function $H s_i - s_i^H + (1 - H)$ equals 0 at $s_i = 1$ and has a positive derivative for $s_i > 1$. Thus $\frac{\partial a_l}{\partial t_{l-1}} > 0$ for all $s_i > 1$. Using this fact, the knowledge that $a_l < \lambda^{-1}(t_l)$, and (A.9), we see that $f(a_l, t_l)$ decreases with increasing $t_{l-1}$, or equivalently it increases with increasing $s_l$. Claim 30 is thus proved.

From (2.33) and (A.7) we obtain that $\cup_t A_t = \mathbb{R}_+$. Exploiting the continuity of $\zeta(s_l, H)$ (see (A.15)) we then have

$$\sup_{b \in \mathbb{R}_+} \zeta(b) = \sup_l \zeta(s_l, H) = \zeta(\sup_l s_l, H) = \zeta(d_\theta, H).$$

(A.17)

Proof of Theorem 3

From (2.35) and the fact that $\zeta(s, H)$ is an increasing function of $s$ (see Claim 30 in the previous proof) we have that $\theta \in \Gamma_{\alpha}$ if and only if $d_\theta \leq \alpha$. Since $d_{\theta_{\alpha, \nu}} = \alpha$ for all $\nu > 0$, we have $\theta_{\alpha, \nu} \in \Gamma(\alpha)$. 
Consider \( \theta = \{ w_k : k \in \mathbb{Z} \} \in \Gamma_\alpha, w_k < w_{k+1} \forall k \), such that \( A_T(\theta) = \min_{\theta \in \Gamma(\alpha)} A_T(\theta) \).

Let \( w_i \) be the first element of \( \theta \) in \((t, \bar{t})\). Set \( \xi = w_i/\alpha^i \). Consider the set \( \theta_{\alpha, \xi} = \{ y_k : y_k = \xi \alpha^k, k \in \mathbb{Z} \} \). Clearly \( y_i = w_i \) and \( y_i \) must be the first element of \( \theta_{\alpha, \xi} \) in \((t, \bar{t})\).

Because \( w_{k+1}/w_k \leq \alpha \) and \( y_{k+1}/y_k = \alpha, \forall k \), we must have \( w_k \leq y_k, \forall k \geq i \). Consequently \( A_T(\theta_{\alpha, \xi}) \leq A_T(\theta) \) which proves (2.41).

It only remains to prove (2.40). We can write \((t, \bar{t})\) as a union of the following \( A_T(\theta_{\alpha, \xi}) + 1 \) intervals: \([y_i, y_i], [y_i, y_{i+1}], [y_{i+1}, y_{i+2}], \ldots, [y_{i+A_T(\theta_{\alpha, \xi})-2}, y_{i+A_T(\theta_{\alpha, \xi})-1}], \) and \([y_{i+A_T(\theta_{\alpha, \xi})-1}, \bar{t}]\). Note that the ratio of the supremum to infimum of each of these intervals is less than or equal to \( \alpha \). Consider \( \theta_{\alpha, \nu} \) for arbitrary \( \nu \). Clearly by definition \( \theta_{\alpha, \nu} \) can have at most one element in each of the these intervals. Thus (2.40) is proved.

Proof of Theorem 4:

The proof relies on the following two claims.

Claim 31 \( L^{[\theta_0]}(b_k) \simeq S^{[\theta_0]}(b_k) \).

Claim 32 \( \lim_{k \to \infty} \frac{C(b_k)}{\mathbb{Q}^*_{\infty}} = 0 \).

From (2.51), (2.50), and Claim 31 we have (2.48). From (2.46) and (A.2) note that

\[ \lambda(b_k) = \lambda^{[\theta_0]}(b_k) = \alpha^k. \tag{A.18} \]

Thus (2.4) and (2.8) give

\[ L^{[\theta_0]}(b_k) = C(b_k), \forall k. \tag{A.19} \]

From (2.43), (2.48), (A.19) and (2.42) we have (2.47). Finally Claim 32 combined with (2.48) gives (2.49).

We now prove the two claims. Recall the definition of \( g(b, t) \) (see (3.4)). For the ease of notation we denote \( g(b_k, \alpha^i) \) by \( g_{k,i} \). From (A.3) and (A.18) we have

\[ \inf_{t > 0} g(b_k, t) = g(b_k, \lambda(b_k)) = g(b_k, \alpha^k) = g_{k,k}. \tag{A.20} \]
Proof of Claim 31: From (2.8), (2.13), (2.28), and (A.20) we have

\[ S^{[\theta,\alpha]}(b_k) = \sum_{l \in \mathbb{Z}} \Phi(g_{k,l}) \]  
(A.21)

and

\[ L^{[\theta,\alpha]}(b_k) = \sup_{l \in \mathbb{Z}} \Phi(g_{k,l}) = \Phi(g_{k,k}). \]  
(A.22)

We now prove that the maximum term, \( \Phi(g_{k,k}) \), dominates the summation of (A.21).

We note two properties of \( \frac{g_{k,l}}{g_{k,k}} \). First, we have from (2.46) and (3.4)

\[
\frac{g_{k,l}}{g_{k,k}} = \frac{b_k + \hat{c} \alpha^l}{\sigma \alpha^H} \cdot \frac{\sigma \alpha^{kH}}{b_k + \hat{c} \alpha^k} \\
= \frac{\alpha^k (1 - H)/H + \alpha^l}{\alpha^H} \cdot \frac{\alpha^{kH}}{\alpha^{k(1 - H)/H + \alpha^k}} \\
= (\alpha^k (1 - H) + H \alpha^l) \cdot \frac{\alpha^{(k-l)H}}{\alpha^k} \\
= (1 - H) \alpha^{(k-l)H} + H \alpha^{(l-k)(1-H)} \\
\geq \epsilon_H \alpha^{|l-k| \epsilon_H},
\]  
(A.23)

where \( \epsilon_H = \min(H, 1 - H) \).

Second, from (A.1) and (A.18) observe that \( \frac{g_{k,l}}{g_{k,k}} \) monotonically increases with increasing \( l \) when \( l \geq k \) and also with decreasing \( l \) when \( l \leq k \). We then have \( \forall l \neq k \)

\[
\frac{g_{k,l}}{g_{k,k}} \geq \min \left( \frac{g_{k,k+1}}{g_{k,k}}, \frac{g_{k,k-1}}{g_{k,k}} \right) \\
= \min((1 - H) \alpha^{-H} + H \alpha^{(1-H)}), (1 - H) \alpha^{H} + H \alpha^{-(1-H)}) \\
=: \mathcal{I}_H > 1
\]  
(A.24)

Now \( g_{k,k} \) is an increasing unbounded function of \( k \):

\[
g_{k,k} = \frac{b_k + \hat{c} \alpha^k}{\sigma \alpha^{kH}} = \frac{\hat{c} \alpha^{k(1-H)}}{\sigma H}.
\]  
(A.25)

Let us assume that \( k \) is large enough to ensure that \( g_{k,k} > 1 \). From page 42 in [8], for \( \delta > 0 \)

\[
\left(1 - \frac{1}{\delta^2}\right) \frac{e^{-\delta^2/2}}{\delta \sqrt{2\pi}} \leq \Phi(\delta) \leq \frac{e^{-\delta^2/2}}{\delta \sqrt{2\pi}}.
\]  
(A.26)
It follows that

\[
\Phi(g_{k,k}) \geq \left( 1 - \frac{1}{g_{k,k}^2} \right) \cdot \frac{e^{-g_{k,k}^2/2}}{g_{k,k} \sqrt{2\pi}} \\
\geq \left( 1 - \frac{1}{g_{k,k}^2} \right) \cdot \frac{e^{-g_{k,k}^2/2}}{g_{k,k}} \cdot \Phi(g_{k,l}). \tag{A.27}
\]

Using (A.23) and (A.24) we have

\[
\Phi(g_{k,l}) \leq \frac{g_{k,l}}{g_{k,k}} \cdot \frac{e^{-(g_{k,l}^2/g_{k,k}^2-1)g_{k,k}^2/2}}{1 - \frac{1}{g_{k,k}^2}} \leq \frac{\alpha^{-|l-k| \epsilon_H}}{\epsilon_H} \cdot \frac{g_{k,k}^2 e^{-((I_H^2-1)g_{k,k}^2/2)}}{g_{k,k}^2 - 1} \tag{A.28}
\]

From (A.21) we have

\[
\Phi(g_{k,k}) \leq S_{\theta k}(b_k) \leq \Phi(g_{k,k}) + \sum_{l>k} \Phi(g_{k,l}) + \sum_{l<k} \Phi(g_{k,l}) \\
\leq \Phi(g_{k,k}) \left( 1 + 2 \cdot \frac{1}{\epsilon_H} \cdot \frac{g_{k,k}^2 e^{-(I_H^2-1)g_{k,k}^2/2}}{g_{k,k}^2 - 1} \sum_{r=1}^{\infty} \frac{\alpha^{-r\epsilon_H}}{\alpha^{-\epsilon_H}} \right) \\
\leq \Phi(g_{k,k}) \left( 1 + 2 \cdot \frac{g_{k,k}^2 e^{-(I_H^2-1)g_{k,k}^2/2}}{g_{k,k}^2 - 1} \cdot \frac{\alpha^{-\epsilon_H}}{1 - \alpha^{-\epsilon_H}} \right). \tag{A.29}
\]

From (A.24) and the fact that \(g_{k,k} \xrightarrow{k \to \infty} \infty\) we have

\[
\lim_{k \to \infty} S_{\theta k}(b_k) = 1, \tag{A.30}
\]

which proves Claim 31.

**Proof of Claim 32:** From (A.26) observe that \(\Phi(\delta) \simeq \frac{e^{-\delta^2/2}}{\delta \sqrt{2\pi}}\).

Set \(\eta := \left( \frac{\tau H}{\sigma H (1-H)^{\frac{3}{2}}} \right)^2\). From (2.30) and (A.3) we then have

\[
C(b) = \Phi(b^{1-H} \eta^{1/2}) \simeq \frac{b^{-1-H}}{\eta^{1/2} \sqrt{2\pi}} e^{-b^2-2H \eta^2}. \tag{A.31}
\]
When $1/2 < H < 1$ we have $0 < \frac{2H-1}{H} < 1$ which implies that
\[
\lim_{b \to \infty} \frac{b^{-(1-H)}}{b^{-(1-H)(2H-1)/H}} = 0.
\] (A.32)

Claim 32 follows from (2.44), (A.31) and (A.32), and the theorem is proved. \qed

The following lemma by Slepian helps us prove Theorem 6.

**Lemma 33** *(page 6 in [69])* Let $\Upsilon[t]$ and $\Psi[t]$, $t \in \theta$, be separable Gaussian random processes, where $\theta$ is a parameter set. If the following relations hold for their covariance functions:
\[
\text{var}(\Upsilon[t]) = \text{var}(\Psi[t]), \quad \forall t \in \theta
\] (A.33)
\[
\text{cov}(\Upsilon[t], \Upsilon[r]) \leq \text{cov}(\Psi[t], \Psi[r]), \quad \forall t, r \in \theta
\] (A.34)
plus their expected values are the same $\forall t$: then for any $x \in \mathbb{R}$
\[
\mathbb{P}\left\{\sup_{t \in \theta} \Upsilon[t] < x\right\} \leq \mathbb{P}\left\{\sup_{t \in \theta} \Psi[t] < x\right\}.
\] (A.35)

**Proof of Lemma 5:** Consider a queue with constant service rate $c$ bits per unit time. Clearly
\[
\sup_{t \in \theta} \mathbb{P}\left\{K[t] - ct > b\right\} \leq \mathbb{P}\left\{\cup_{t \in \theta} \{K[t] - ct > b\}\right\} \leq \sum_{t \in \theta} \mathbb{P}\{K[t] - ct > b\}.
\] (A.36)

From (2.6) we see that $\mathbb{P}\{Q[0] > b\}$ is identical to $\mathbb{P}\left\{\cup_{t \in \theta} \{K[t] - ct > b\}\right\}$. Then (2.8), (2.13) and (A.36) give (2.50).

We now prove that for $0 \leq a_k \leq 1$, $k = 1, \ldots, l$,
\[
\max_{k=1, \ldots, l} (1 - a_k) \leq 1 - \prod_{k=1}^{l} a_k \leq \sum_{k=1}^{l} (1 - a_k),
\] (A.37)
which is equivalent to (2.51). The first inequality in (A.37) is trivial. We prove the second inequality in (A.37) by induction. Assuming that
\[
1 - \prod_{k=1}^{l-1} a_k \leq \sum_{k=1}^{l-1} (1 - a_k),
\] (A.38)
we have
\[
\prod_{k=1}^{l} a_k \geq \left( 1 - \sum_{k=1}^{l-1} (1 - a_k) \right) a_l = a_l - a_l \sum_{k=1}^{l-1} (1 - a_k) \geq 1 - (1 - a_l) - \sum_{k=1}^{l-1} (1 - a_k) = 1 - \sum_{k=1}^{l} (1 - a_k).
\]  
(A.39)

The initial induction step, that is (A.38) for \( l = 2 \), is trivial thus proving the lemma.

\[\square\]

**Lemma 34** *(see [14]):* If \( E_1 \subset E_2 \cdots \) and \( E = \cup_i E_i \) then \( \lim_{i \to \infty} \mathbb{P} \{ E_i \} = \mathbb{P} \{ E \} \). If \( E_1 \supset E_2 \cdots \) and \( E = \cap_i E_i \) then \( \lim_{i \to \infty} \mathbb{P} \{ E_i \} = \mathbb{P} \{ E \} \).

**Proof of Theorem 6**

Define independent Gaussian random variables
\[
\Upsilon[t] \sim \mathcal{N}(\mathbb{E}(K_r[t] - tc), \text{var}(K_r[t])), \ t \in \theta.
\]  
(A.40)

and set
\[
\Psi[t] := K_r[t] - tc, \ t \in \theta.
\]  
(A.41)

From our assumption that \( \text{cov}(K_r[t], K_r[r]) \geq 0 \) we see that \( \Upsilon[t] \) and \( \Psi[t] \) satisfy the conditions in Lemma 33 and so (A.35) holds. Note that
\[
\mathbb{P} \left\{ \sup_{t \in \theta} \Psi[t] < b \right\} = 1 - \mathbb{P} \left\{ Q_r^{[\theta]} > b \right\}.
\]  
(A.42)

Label the elements of \( \theta \) as \( \{ t_k \}_{k \in \mathbb{Z}} \), which we assume satisfy (2.33). Define the sets
\[
E_i := \cap_{k=-i}^{i} \{ \Upsilon[t_k] < b \} \quad \text{and} \quad E := \cap_i E_i = \{ \sup_{t \in \theta} \Upsilon[t] < b \}.
\]  
Then from Lemma 34
we have \( \lim_{t \to \infty} \mathbb{P} \{ E_t \} = \mathbb{P} \{ E \} \) and consequently

\[
P^{[\theta]}(b) = 1 - \prod_{t \in \theta} \mathbb{P} \{ \Upsilon[t] < b \} = 1 - \mathbb{P} \left\{ \sup_{t \in \theta} \Upsilon[t] < b \right\},
\]

(A.43)

Then from (A.35), (A.42) and (A.43) the theorem is proved. \( \square \)

**Proof of Theorem 9:** A basic result from analysis [9] states that \( \sum_{k=1}^{\infty} \log(1 - a_k) > -\infty \), that is \( \prod_{k=1}^{\infty} (1 - a_k) > 0 \), if and only if \( \sum_{k=1}^{\infty} a_k < \infty \) where \( 0 \leq a_k < 1 \).

We thus only need to show that \( S^{[\theta, \alpha]}(b) < \infty \) to prove the theorem. Recall that \( b_k = \alpha^k \tilde{c}(1 - H)/H \) and \( g_{k, l} = \frac{b_{k+l} - \tilde{c} \alpha^l}{\sigma \alpha^l \Gamma} \). From (A.21), (A.23), (A.24), and (A.26) we have

\[
S^{[\theta, \alpha]}(b_k) = \Phi(g_{k, k}) + \sum_{l=k+1}^{\infty} \Phi(g_{k, l}) + \sum_{l=-\infty}^{k-1} \Phi(g_{k, l})
\]

\[
\leq \Phi(g_{k, k}) + \sum_{l=k+1}^{\infty} \frac{e^{-g_{k, l}^2/2}}{g_{k, l} \sqrt{2\pi}} + \sum_{l=-\infty}^{k-1} \frac{e^{-g_{k, l}^2/2}}{g_{k, l} \sqrt{2\pi}}
\]

\[
\leq \Phi(g_{k, k}) + \sum_{l=k+1}^{\infty} \frac{e^{-g_{k, l}^2/2}}{g_{k, l} \sqrt{2\pi}} + \sum_{l=-\infty}^{k-1} \frac{e^{-g_{k, l}^2/2}}{g_{k, l} \sqrt{2\pi}}
\]

\[
\leq \Phi(g_{k, k}) + \frac{e^{-g_{k, k}^2/2}}{g_{k, k} \epsilon H \sqrt{2\pi}} \left( \sum_{l=k+1}^{\infty} \alpha^{-|l-k|H} + \sum_{l=-\infty}^{k-1} \alpha^{-|l-k|H} \right)
\]

\[
= \Phi(g_{k, k}) + \frac{e^{-g_{k, k}^2/2}}{g_{k, k} \epsilon H \sqrt{2\pi}} \cdot \frac{2 \alpha^{-H}}{1 - \alpha^{-H}}
\]

\[
< \infty.
\]

(A.44)

Because \( \mathbb{P} \{ K[t] - ct > b \} \forall t \), is a decreasing function of \( b \), so is \( S^{[\theta, \alpha]}(b) \) (see (2.13)).

Since for any given \( b > 0 \) we can find \( k \) such that \( b_k < b \) we have \( S^{[\theta, \alpha]}(b) < \infty \forall b \), and the theorem is proved. \( \square \)
Appendix B

Proofs of WIG and MWM fed queues

The following lemma helps prove Theorem 8.

Lemma 35 (Lemma 6 of [72]) Define the distance between two WIG or MWM multiscale tree leaf nodes as the depth of their lowest common ancestor. Let $C_l$ denote the covariance of two leaf nodes with distance $l$. If $C_{l+1} \geq C_l$ for all $l$, then

$$\text{var}(K_{\text{end}}[2^i]) \geq \text{var} \left( \sum_{k \in B_i} V_{n,k} \right)$$

(B.1)

where $i = 0, \ldots, n$ and $B_i$ is any subset of $\{0, \ldots, 2^n - 1\}$ of size $2^i$.

Proof of Theorem 8: We start with preliminaries regarding the covariance structure of WIG tree leaf nodes. The leaf node $V_{n,k}$ (see Fig. 2.1) has the representation

$$V_{n,k} = 2^{-n}V_{0,0} + \sum_{j=0}^{n-1} (-1)^{k_j+1}2^{-(n-j)}Z_{j,k}$$

(B.2)

where $k_j = \lfloor k/2^n \rfloor$. Any leaf node $V_{n,k'}$ with distance $l (\leq n - 1)$ from $V_{n,k}$ has the representation

$$V_{n,k'} = 2^{-n}V_{0,0} + \sum_{j=0}^{l-1} (-1)^{k_j+1}2^{-(n-j)}Z_{j,k} - (-1)^{k_l+1}2^{-(n-l)}Z_{l,k_l} + \sum_{j=l+1}^{n-1} (-1)^{k_j'+1}2^{-(n-j)}Z_{j,k_j'}$$

(B.3)

where $k_j' \neq k_j$ for $j > l$. For the ease of notation we denote $\text{var}(Z_{j,k})$ by $\text{var}(Z_j)$ because it is not a function of $k$. Since $V_{0,0}$ and the $Z_{j,k}$’s are independent we have

$$C_l = \text{cov}(V_{n,k}, V_{n,k'})$$
\begin{equation}
2^{-2n}\text{var}(V_{0,0}) + \sum_{j=0}^{l-1} 2^{-2(n-j)}\text{var}(Z_j) - 2^{-2(n-l)}\text{var}(Z_l). \tag{B.4}
\end{equation}

Using (2.56) and (B.4) we have for $0 \leq l \leq n - 2$

\begin{align*}
C_{l+1} - C_l &= 2 \cdot 2^{-2(n-l)}\text{var}(Z_l) - 2^{-2(n-l-1)}\text{var}(Z_{l+1}) \\
&= 2^{-2(n-l)+1}(\text{var}(Z_l) - 2\text{var}(Z_{l+1})) \\
&\geq 0. \tag{B.5}
\end{align*}

Now from (B.2) and (B.4) we have

\begin{equation}
C_n = \text{cov}(V_{n,k}, V_{n,k}) = 2^{-2n}\text{var}(V_{0,0}) + \sum_{j=0}^{n-1} 2^{-2(n-j)}\text{var}(Z_j) \geq C_l, \quad \forall l \leq n. \tag{B.6}
\end{equation}

Thus $C_l$ increases with distance $l$. From (2.56), (B.5), and (B.6) we also have for $0 \leq l \leq n$ that

\begin{equation}
C_l \geq C_0 = 2^{-2n}\text{var}(V_{0,0}) - 2^{-2n}\text{var}(Z_{0,0}) \geq 0. \tag{B.7}
\end{equation}

We use (B.5) and (B.7) to prove the two following claims.

**Claim 36** $\text{var}(K_{\text{end}}[2^i]) \geq \text{var}(K_{\tau}[2^i])$ for $i = 0, \ldots, \lfloor \log_2 \tau \rfloor$ and for $\tau = 1, \ldots, 2^n$.

**Claim 37** $\text{cov}(K_{\tau}[t], K_{\tau}[r]) \geq 0$ for $0 \leq t, r \leq \tau$.

Note that $K_{\tau}[2^i]$ is one possible value of $\sum_{k \in B_i} V_{n,k}$ in (B.1). Thus Claim 36 follows from Lemma 35 and (B.5). We note that Claim 36 was proved specifically for a WIG model of fGn in [53] (Theorem 5 therein). A minor modification in the analysis of [53] however generalizes the result for all model parameters satisfying (2.56).

Because $K_{\tau}[t] = \sum_{k=\tau-t}^{\tau-1} V_{n,k}$ it follows that $\text{cov}(K_{\tau}[t], K_{\tau}[r])$ is a linear combination of the $C_l$’s with \textit{positive} weights. Claim 37 then follows from this fact and from (B.7).
Note that
\[ P\left\{ K_\tau[t] - \tilde{c}(n)t < b \right\} = 1 - \Phi \left( \frac{b + \tilde{c}(n)t - \mathbb{E}(K_\tau[t])}{\sqrt{\text{var}(K_\tau[t])}} \right). \] (B.8)

From (2.26) we have that
\[ b + \tilde{c}(n)t - \mathbb{E}(K_\tau[t]) > 0. \] (B.9)

Since the process \( V_{n,k} \) is first-order stationary, \( \mathbb{E}(K_{\text{end}}[t]) = \mathbb{E}(K_\tau[t]) \) for all \( \tau \) and \( t \).

This fact along with Claim 36, (B.8), and (B.9) then give
\[ P\left\{ K_\tau[2^i] - \tilde{c}(n)2^i < b \right\} \leq P\left\{ K_{\text{end}}[2^i] - \tilde{c}(n)2^i < b \right\}, \; i = 0, \ldots, \lfloor \log_2 \tau \rfloor, \; \tau = 1, \ldots, 2^n. \] (B.10)

We thus have
\[ P_{\text{end}}^{[\theta_2]}(b) \geq P_\tau^{[\theta_2]}(b), \; \forall \tau = 1, \ldots, 2^n. \] (B.11)

Claim 37 and Theorem 6 give
\[ P_\tau^{[\theta_2]}(b) \geq \mathbb{P}\left\{ Q_\tau^{[\theta_2]} > b \right\}, \; \forall \tau = 1, \ldots, 2^n. \] (B.12)

Combining (B.11) and (B.12) proves the theorem. \( \square \)

**Lemma 38** Assume that the events \( W_i \) are of the form \( W_i = \{ I_i < \kappa_i \} \), where \( I_i = R_0 + R_1 + \ldots + R_i \) for \( 1 \leq i \leq n \) and where \( R_0, \ldots, R_n \) are independent, otherwise arbitrary random variables. Then, for \( 1 \leq i \leq n \), we have
\[ \mathbb{P}\left\{ W_i | W_{i-1}, \ldots, W_0 \right\} \geq \mathbb{P}\left\{ W_i \right\}. \] (B.13)

**Proof:** We first spell out some notation. By \( f_L \) and \( F_L \) we denote the probability density function (PDF) and cumulative distribution function (CDF), respectively, of a random variable \( L \). Furthermore, we denote by \( F_L|E(l) \) the CDF of \( L \) conditioned on knowing
the event \( E \). For convenience, let us write \( W_i := \{ I_i < \kappa_i \} \) for short, and let us introduce
the auxiliary random variables \( Y_0 := L_0 := I_0 := R_0, \)

\[
Y_i := I_i | W_{i-1}, \ldots, W_0 \quad \text{and} \quad L_i := I_i | W_i, \ldots, W_0, \quad i \geq 1. \tag{B.14}
\]

To prove the lemma, it is enough to show that

\[
F_{Y_i}(r) \geq F_{L_i}(r) \tag{B.15}
\]

\( \forall r \in \mathbb{R} \) and \( \forall i \) and then set \( r = \kappa_i \).

We prove (B.15) by induction. First note that \( F_{Y_0}(r) \geq F_{I_0}(r) \). Next, we assume
that (B.15) holds for \( i \) and show that it holds also for \( i + 1 \). Bayes’ rule yields

\[
F_{L_i}(r) = \begin{cases}
\frac{F_{Y_i}(r)}{F_{Y_i}(\kappa_i)}, & \text{if } r \leq \kappa_i \\
1, & \text{otherwise}
\end{cases} \geq F_{Y_i}(r). \tag{B.16}
\]

The key to the proof is to note that \( Y_{i+1} = L_i + R_{i+1} \), where \( R_{i+1} \) is independent of \( I_j \)
and hence of \( W_j \) for \( j \leq i \). In short, \( R_{i+1} \) is independent of \( L_i \). This fact, (B.15) and
(B.16) allow us to write

\[
F_{Y_{i+1}}(r) = \mathbb{P}\{L_i + R_{i+1} < r\} = \int_{-\infty}^{\infty} \int_{-\infty}^{r-r_{i+1}} f_{L_i}(l_i) f_{R_{i+1}}(r_{i+1}) \, dl_i \, dr_{i+1}
\]

\[
= \int_{-\infty}^{\infty} F_{L_i}(r - r_{i+1}) f_{R_{i+1}}(r_{i+1}) \, dr_{i+1}
\]

\[
\geq \int_{-\infty}^{\infty} F_{Y_i}(r - r_{i+1}) f_{R_{i+1}}(r_{i+1}) \, dr_{i+1}
\]

\[
\geq \int_{-\infty}^{\infty} F_{I_i}(r - r_{i+1}) f_{R_{i+1}}(r_{i+1}) \, dr_{i+1}
\]

\[
= \mathbb{P}\{I_i + R_{i+1} < r\} = F_{I_{i+1}}(r). \tag{B.17}
\]

This proves the claim by induction. \( \square \)
Proof of Theorem 7

Let us first show that Lemma 38 applies to the WIG and the MWM for the events \( W_i = K_{\text{end}}[2^{n-i}] < b \). To this end we need only show that these \( W_i \) can be written in the appropriate form. Recall that we have \( K_{\text{end}}[2^{n-i}] = V_{i,2^{i-1}} \).

**WIG:** The WIG uses additive innovations \( Z_{j,k} \) arranged on a tree as in Fig. 2.1. It is immediate from (2.18) that \( K_{\text{end}}[2^{n-i}] \) becomes

\[
K_{\text{end}}[2^{n-i}] = V_{i,2^{i-1}} = 2^{-i}V_{0,0} - \sum_{j=0}^{i-1} 2^{j-i}Z_{j,2^{j-1}}. 
\]

(B.18)

It suffices, thus, to set \( \kappa_i = 2^ib + 2^{n-i}\tilde{c}(n) \), \( R_0 = V_{0,0} \) and \( R_i = -2^{i-1}Z_{i-1,2^{i-1}-1} \).

**MWM:** The MWM employs the same tree structure as the WIG, however, with multiplicative innovations \( U_{j,k} \). Recalling (2.20), \( K_{\text{end}}[2^{n-i}] \) becomes

\[
K_{\text{end}}[2^{n-i}] = V_{i,2^{i-1}} = V_{0,0} \prod_{j=0}^{i-1} (1 - U_j). 
\]

(B.19)

Taking logarithms, it is a simple task to write the events \( W_i \) in the required form, this time by setting \( \kappa_i = \ln(b + 2^{n-i}\tilde{c}(n)) \), \( R_0 = \ln(V_{0,0}) \), and \( R_i = \ln(1 - U_{i-1}) \).

Using (B.13) we find

\[
\mathbb{P}\left\{Q_{\text{end}}^{[b_2]} > b \right\} = 1 - \mathbb{P}\left\{Q_{\text{end}}^{[b_2]} < b \right\} = 1 - \mathbb{P}\{\bigcap_{i=0}^{n} W_i \}
\]

\[
= 1 - \mathbb{P}\{W_0\} \prod_{i=1}^{n} \mathbb{P}\{W_i | W_{i-1}, \ldots, W_0\}
\]

\[
\leq 1 - \prod_{i=0}^{n} \mathbb{P}\{W_i\} = P_{\text{end}}^{[b_2]}(b). 
\]

(B.20)

Proof of Theorem 10

Let us use the superscript \((n)\) to denote variables corresponding to a tree of depth \( n \).

Define the events

\[
W_j^{(n)} = \{K^{(n)}[2^{n-j}] < b + \tilde{c}^{(n)}2^{n-j}\}. 
\]

(B.21)
We require the notion of a threshold scale $N$ to prove Theorem 10: Let $N$ be such that
\[ \mathbb{P} \left\{ W^{(j)}_j \right\} \geq 1 - 2^{-j}, \ \forall j \geq N. \] (B.22)

We start by showing the existence of such a threshold scale.
\[ \mathbb{P} \left\{ W^{(n)}_n \right\} = \mathbb{P} \left\{ \varrho U_{-1} \ldots U_{n-1} < b + \bar{c}^{(n)} \right\}. \] (B.23)

For short, let us write $\Lambda_i := \log_2(U_i)$ and
\[ \chi_n := -\frac{1}{n} \log_2(b/\varrho + \bar{c}^{(n)}/\varrho). \] (B.24)

For any $r > 0$, a simple application of the Jensen inequality and independence yields the Chernoff bound:
\[ 1 - \mathbb{P} \left\{ W^{(n)}_n \right\} = \mathbb{P} \left\{ -(1/n)(\Lambda_{-1} + \ldots + \Lambda_{n-1}) < \chi_n \right\} \]
\[ = \mathbb{P} \left\{ 2^{r(\Lambda_{-1} + \ldots + \Lambda_{n-1})} > 2^{-rn\chi_n} \right\} \]
\[ \leq \frac{\mathbb{E}(2^{r(\Lambda_{-1} + \ldots + \Lambda_{n-1})})}{2^{-rn\chi_n}} \]
\[ = 2^{n(-T^{(n)}(r) - 1 + r\chi_n)}. \] (B.25)

Here we set
\[ T^{(n)}(r) = -1 - (1/n) \sum_{i=-1}^{n-1} \log_2 \mathbb{E}(U_i^r). \] (B.26)

Now, taking logarithms and minimizing over $r > 0$ yields
\[ (1/n) \log_2 \left( 1 - \mathbb{P} \left\{ W^{(n)}_n \right\} \right) \leq \inf_{r > 0} \left( r\chi_n - T^{(n)}(r) \right) - 1 \]
\[ = -1 + (T^{(n)})^*(\chi_n) \leq -1 \] (B.27)
provided that $\chi_n$ is small enough that $(T^{(n)})^*(\chi_n) < 0$. Here $(T^{(n)})^*$ is the Legendre transform of $T^{(n)}$ defined by the infimum expression of (B.27). Now $(T^{(n)})^*$ converges to a concave function $T^*$ that has two zeros, the smaller zero being strictly positive but
smaller than 1 [77]. Also, we point out that $\chi_n$ decreases to 0 and that $T^{(n)}$ converges to $T$, implying that the zero of $(T^{(n)})^*$ will not change greatly once $n$ is large.

We can thus assume that $(T^{(n)})^*(\chi_n)$ is negative for all $n$ greater than or equal to some critical $N$, which is the condition needed for completing the proof rigorously. For $n \geq N$, we have then $1 - P[W^{(n)}_n] \leq 2^{-n} \leq 2^{-N}$ which proves (B.22). Choosing $\varepsilon = \log_2(1 - 2^{-N})/(2^{-N}) < 0$, we guarantee that $\log_2 P[W^{(n)}_n] \geq \varepsilon \left(1 - P[W^{(n)}_n]\right)$ for all $n \geq N$. We conclude that

$$\log_2 \prod_{n=N}^{N'} P[W^{(n)}_n] \geq \varepsilon \sum_{n=N}^{N'} \left(1 - P[W^{(n)}_n]\right)$$

$$\geq \varepsilon \sum_{n=N}^{N'} 2^{-n} \geq \varepsilon \sum_{n=N}^{\infty} 2^{-n}$$

$$\geq \varepsilon 2^{-N+1}. \quad \text{(B.29)}$$

Thus we may estimate the “neglected terms” in $P^{(N)}(b)$ by $1 \geq \prod_{n=N}^{\infty} P[W^{(n)}_n] \geq 2^\varepsilon 2^{-N+1}$, which leads to

$$1 - P^{(N)}(b) \geq 1 - P^{(\infty)}(b)$$

$$\geq \left(1 - P^{(N)}(b)\right) 2^\varepsilon 2^{-N+1}$$

$$= \left(1 - P^{(N)}(b)\right) (1 - 2^{-N})^2. \quad \text{(B.30)}$$

Our choice of a $\beta$-distribution [46] for the tree-root $V_{0,0}$ and innovations $U_{j,k}$ (see (2.22) and (2.21)) ensures that $P\{\mathcal{K}^{(n)}[2^i] - tc < b\} > 0$ for all $i \in \{0, \ldots, n\}$ and for all tree depths $n$. Thus

$$P^{(N)}(b) < 1. \quad \text{(B.31)}$$

From (B.30) and (B.31) we have (2.66). □
Appendix C

Proofs for independent innovations trees

Proof of (3.11) (Adapted from Appendix A in [21])

Without loss of generality assume that $V_\gamma$ has mean zero. We have

$$L_{\gamma k} = H_k V_\gamma + W_k,$$

where $H_k$ is a constant vector and $W_k$ is a function of random innovations. Defining

$$H := [H^T_0, H^T_1, \ldots, H^T_{P,}]^T$$

(C.2)

and

$$W := [W^T_0, W^T_1, \ldots, W^T_{P,}]^T$$

(C.3)

we have

$$L_\gamma = H V_\gamma + W.$$

(C.4)

Denote $\text{var}(V_\gamma)$ by $v$. It follows that

$$S_{L_\gamma} = HH^T v + S_W$$

(C.5)

and

$$\text{cov}(L_\gamma, V_\gamma) = Hv.$$  

(C.6)
Using (3.17), (C.5), and (C.6) we obtain

\[
\mathcal{E}(V_{\gamma}|L_{\gamma})(\frac{1}{v} + \mathcal{H}^T S_W^{-1} \mathcal{H})
\]

\[
= (v - (\mathcal{H}^T v) S_{L_{\gamma}}^{-1}(\mathcal{H} v))(\frac{1}{v} + \mathcal{H}^T S_W^{-1} \mathcal{H})
\]

\[
= 1 - \mathcal{H}^T S_{L_{\gamma}}^{-1} \mathcal{H} v + \mathcal{H}^T S_W^{-1} \mathcal{H} v - (\mathcal{H}^T v) S_{L_{\gamma}}^{-1}(\mathcal{H} v) \mathcal{H}^T S_W^{-1} \mathcal{H}
\]

\[
= 1 - \mathcal{H}^T S_{L_{\gamma}}^{-1} \mathcal{H} v + \mathcal{H}^T S_W^{-1} \mathcal{H} v - (\mathcal{H}^T v) S_{L_{\gamma}}^{-1}(S_{L_{\gamma}} - S_W) S_W^{-1} \mathcal{H}
\]

\[
= 1.
\]

(C.7)

Key to the proof is the fact that the vectors $W_k$, $k = 1, 2, \ldots, P_{\gamma}$ are independent. This along with (C.7) completes the proof as follows.

\[
\frac{1}{\mathcal{E}(V_{\gamma}|L_{\gamma})} = \frac{1}{v} + \mathcal{H}^T S_W^{-1} \mathcal{H}
\]

\[
= \frac{1}{v} + \sum_{k=1}^{P_{\gamma}} \mathcal{H}_k^T S_W^{-1} \mathcal{H}_k
\]

\[
= \frac{1}{v} + \sum_{k=1}^{P_{\gamma}} \left(\frac{1}{\mathcal{E}(V_{\gamma}|L_{\gamma})} - \frac{1}{v}\right).
\]

(C.8)

Proof of Lemma 18

We first prove the following statement.

Claim 39 If there exists $X^* = [x_k^*] \in \Delta_n(M_1, \ldots, M_P)$ that has the following property:

\[
\psi_i(x_i^*) - \psi_i(x_i^* - 1) \geq \psi_j(x_j^* + 1) - \psi_j(x_j^*),
\]

(C.9)

\[\forall i \neq j \text{ such that } x_i^* > 0 \text{ and } x_j^* < M_j, \text{ then}\]

\[
h(n) = \sum_{k=1}^{P} \psi_k(x_k^*).\]

(C.10)

We then prove that such an $X^*$ always exists and can be constructed using the water-filling technique.
Consider any \( \hat{X} \in \Delta_n(M_1, \ldots, M_P) \). Using the following steps, we transform the vector \( \hat{X} \) two elements at a time to obtain \( X^* \).

**Step 1:** (Initialization) Set \( X = \hat{X} \).

**Step 2:** If \( X \neq X^* \) then there exists a pair \( i, j \) such that \( x_i \neq x_i^* \) and \( x_j \neq x_j^* \) since the elements of both \( X \) and \( X^* \) sum up to \( n \). Without loss of generality assume that \( x_i < x_i^* \) and \( x_j > x_j^* \). This assumption implies that \( x_i^* > 0 \) and \( x_j^* < M_j \). Now form vector \( Y \) such that

\[
\begin{align*}
y_i &= x_i + 1 \\
y_j &= x_j - 1 \\
y_k &= x_k, \ (k \neq i, j).
\end{align*}
\] (C.11)

From (C.9) and the concavity of \( \psi_i \) and \( \psi_j \) we have

\[
\psi_i(y_i) - \psi_i(x_i) = \psi_i(x_i + 1) - \psi_i(x_i) \\
\geq \psi_i(x_i^*) - \psi_i(x_i^* - 1) \\
\geq \psi_j(x_j^* + 1) - \psi_j(x_j^*) \\
\geq \psi_j(x_j) - \psi_j(x_j - 1) \\
\geq \psi_j(x_j) - \psi_j(y_j).
\] (C.12)

As a consequence

\[
\sum_k (\psi_k(y_k) - \psi_k(x_k)) = \psi_i(y_i) - \psi_i(x_i) + \psi_j(y_j) - \psi_j(x_j) \\
\geq 0.
\] (C.13)

**Step 3:** If \( Y \neq X^* \) then set \( X = Y \) and repeat Step 2, otherwise stop.

After performing the above steps at most \( \sum_k M_k \) times, \( Y = X^* \) and (C.13) gives

\[
\sum_k \psi_k(x_k^*) = \sum_k \psi_k(y_k) \geq \sum_k \psi_k(\hat{x}_k).
\] (C.14)

This proves Claim 39.
Indeed for any $\tilde{X} \neq X^*$ satisfying (C.9) we must have $\sum_k \psi_k(\tilde{x}_k) = \sum_k \psi_k(x_k^*)$.

We now prove the following claim by induction.

**Claim 40** $G^{(n)} \in \Delta_n(M_1, \ldots, M_P)$ and that $G^{(n)}$ satisfies (C.9).

(Initial Condition) The claim is trivial for $n = 0$.

(Induction Step) Clearly from (3.6) and (3.7)

$$\sum_k g_k^{(n+1)} = 1 + \sum_k g_k^{(n)} = n + 1,$$

(C.15)

and $0 \leq g_k^{(n+1)} \leq M_k$. Thus $G^{(n+1)} \in \Delta_{n+1}(M_1, \ldots, M_P)$. We now prove that $G^{(n+1)}$ satisfies property (C.9). We need to consider pairs $i, j$ as in (C.9) for which either $i = m$ or $j = m$ because all other cases directly follow from the fact that $G^{(n)}$ satisfies (C.9).

Case (i) $j = m$, where $m$ is defined as in (3.7). Assuming that $g_m^{(n+1)} < M_m$, for all $i \neq m$ such that $g_i^{(n+1)} > 0$ we have

$$\psi_i \left( g_i^{(n+1)} \right) - \psi_i \left( g_i^{(n+1)} - 1 \right)$$

$$= \psi_i \left( g_i^{(n)} \right) - \psi_i \left( g_i^{(n)} - 1 \right)$$

$$\geq \psi_m \left( g_m^{(n)} + 1 \right) - \psi_m \left( g_m^{(n)} \right)$$

$$\geq \psi_m \left( g_m^{(n)} + 2 \right) - \psi_m \left( g_m^{(n)} + 1 \right)$$

$$= \psi_m \left( g_m^{(n+1)} + 1 \right) - \psi_m \left( g_m^{(n+1)} \right).$$

(C.16)

Case (ii) $i = m$. Consider $j \neq m$ such that $g_j^{(n+1)} < M_j$. We have from (3.7) that

$$\psi_m \left( g_m^{(n+1)} \right) - \psi_m \left( g_m^{(n+1)} - 1 \right)$$

$$= \psi_m \left( g_m^{(n)} + 1 \right) - \psi_m \left( g_m^{(n)} \right)$$

$$\geq \psi_j \left( g_j^{(n)} + 1 \right) - \psi_j \left( g_j^{(n)} \right)$$

$$= \psi_j \left( g_j^{(n+1)} + 1 \right) - \psi_j \left( g_j^{(n+1)} \right).$$

(C.17)
Thus Claim 40 is proved.

It only remains to prove the next claim.

**Claim 41** $h(n)$, or equivalently $\sum_k \psi_k(g_k^{(n)})$, is non-decreasing and concave.

Since $\psi_k$ is non-decreasing for all $k$, from (3.6) we have that $\sum_k \psi_k(g_k^{(n)})$ is a non-decreasing function of $n$. We have from (3.7)

$$h(n+1) - h(n) = \sum_k \left( \psi_k(g_k^{(n+1)}) - \psi_k(g_k^{(n)}) \right)$$

$$= \max_{k: g_k^{(n)} < M_k} \left\{ \psi_k(g_k^{(n)} + 1) - \psi_k(g_k^{(n)}) \right\}.$$  \hspace{1cm} (C.18)

From the concavity of $\psi_k$ and the fact that $g_k^{(n+1)} \geq g_k^{(n)}$ we have that

$$\psi_k(g_k^{(n)} + 1) - \psi_k(g_k^{(n)}) \geq \psi_k(g_k^{(n+1)} + 1) - \psi_k(g_k^{(n+1)}),$$  \hspace{1cm} (C.19)

for all $k$. Thus from (C.18) and (C.19), $h(n)$ is concave.

**Proof of Corollary 19**

Set $x_k^* = \lfloor \frac{n}{P} \rfloor$ for $1 \leq k \leq P - n + P \left\lfloor \frac{n}{P} \right\rfloor$ and $x_k^* = 1 + \lfloor \frac{n}{P} \rfloor$ for all other $k$. Then $X^* = [x_k^*] \in \Delta_n(M_1, \ldots, M_P)$ and $X^*$ satisfies (C.9) from which the result follows.

The following two lemmas are required to prove Theorem 20.

**Lemma 42** Given independent random variables $A, W, F$, define $B$ and $C$ through $B := \zeta A + W$ and $C := \eta B + F$ where $\zeta, \eta$ are constants. We then have the result

$$\frac{\text{var}(A)}{\text{cov}(A, C)^2} - \frac{\text{cov}(B, C)^2}{\text{var}(B)} = \frac{\zeta^2 + \text{var}(W)/\text{var}(A)}{\zeta^2} \geq 1.$$  \hspace{1cm} (C.20)
Proof: Without loss of generality assume all random variables have zero mean. We have

\[
\text{cov}(C, B) = \mathbb{E}(\eta B^2 + FB) = \eta \text{var}(B), \tag{C.21}
\]

\[
\text{cov}(A, C) = \mathbb{E}(\eta(\zeta A + W + F)A) = \mathbb{E}(\zeta \eta A^2 + \eta WA + FA) = \zeta \eta \text{var}(A), \tag{C.22}
\]

and

\[
\text{var}(B) = \mathbb{E}(\zeta^2 A^2 + W^2 + 2\zeta AW) = \zeta^2 \text{var}(A) + \text{var}(W). \tag{C.23}
\]

Thus from (C.21), (C.22) and (C.23)

\[
\frac{\text{cov}(B, C)^2}{\text{var}(B)} \cdot \frac{\text{var}(A)}{\text{cov}(A, C)^2} = \frac{\eta^2 \text{var}(B)}{\zeta^2 \eta^2 \text{var}(A)} = \frac{\zeta^2 + \text{var}(W) / \text{var}(A)}{\zeta^2} \geq 1. \tag{C.24}
\]

\(\square\)

Lemma 43 Given a positive function \(\rho_i, i \in \mathbb{Z}\) and constant \(\alpha > 0\) such that

\[
q_i := \frac{1}{1 - \beta \rho_i} \tag{C.26}
\]

is also positive, concave, and non-decreasing for all \(\beta\) such that \(0 < \beta \leq \alpha\).
Proof: Define \( \kappa_i := \rho_i - \rho_{i-1} \). Since \( \rho_i \) is positive and \( r_i \) is positive and non-decreasing, \( \alpha \rho_i < 1 \) and \( \rho_i \) must increase with \( i \), that is \( \kappa_i \geq 0 \). This combined with the fact that \( \beta \rho_i \leq \alpha \rho_i < 1 \) guarantees that \( q_i \) must be positive and non-decreasing.

It only remains to prove the concavity of \( q_i \). From (C.25)

\[
\frac{r_{i+1} - r_i}{r_{i+1} - r_i} = \frac{1}{1 - \alpha \rho_{i+1}} - \frac{1}{1 - \alpha \rho_i} = \frac{\alpha (\rho_{i+1} - \rho_i)}{(1 - \alpha \rho_{i+1})(1 - \alpha \rho_i)} = \alpha \kappa_{i+1} \frac{r_{i+1} r_i}{r_{i+1} r_i}.
\]

We are given that \( r_i \) is concave, that is

\[
0 \geq (r_{i+2} - r_{i+1}) - (r_{i+1} - r_i) = \alpha r_{i+1} \left[ \kappa_{i+2} \left( \frac{1 - \alpha \rho_i}{1 - \alpha \rho_{i+2}} \right) - \kappa_{i+1} \right]. \tag{C.28}
\]

Since \( r_i > 0 \ \forall i \), we must have

\[
\left[ \kappa_{i+2} \left( \frac{1 - \alpha \rho_i}{1 - \alpha \rho_{i+2}} \right) - \kappa_{i+1} \right] \leq 0. \tag{C.29}
\]

Similar to (C.28) we have that

\[
(g_{i+2} - g_{i+1}) - (g_{i+1} - g_i) = \beta q_i q_{i+1} \left[ \kappa_{i+2} \left( \frac{1 - \beta \rho_i}{1 - \beta \rho_{i+2}} \right) - \kappa_{i+1} \right]. \tag{C.30}
\]

Since \( q_i > 0 \ \forall i \), for the concavity of \( q_i \) it suffices to show

\[
\left[ \kappa_{i+2} \frac{1 - \beta \rho_i}{1 - \beta \rho_{i+2}} - \kappa_{i+1} \right] \leq 0. \tag{C.31}
\]

Now

\[
\frac{1 - \alpha \rho_i}{1 - \alpha \rho_{i+2}} = \frac{1 - \beta \rho_i}{1 - \beta \rho_{i+2}} = \frac{(\alpha - \beta)(\rho_{i+2} - \rho_i)}{(1 - \alpha \rho_{i+2})(1 - \beta \rho_{i+2})} \geq 0. \tag{C.32}
\]
Then (C.29) and (C.32) combined with the fact that $\kappa_i \geq 0$, $\forall i$ proves (C.31).

**Proof of Theorem 20**

We split the theorem into three claims.

**Claim 44** $L^* := \bigcup_k L^{(k)}(x_k^*) \in \mathcal{L}_\gamma(n)$.

From (3.11), (3.12), and (3.14) we obtain

$$
\mu_\gamma(n) + \frac{P_\gamma - 1}{\text{var}(V_\gamma)} = \max_{L \in \Lambda_\gamma(n)} \sum_{k=1}^{P_\gamma} \mathcal{E}^{-1}(V_\gamma | L_{\gamma k}) \\
\leq \max_{x \in \Delta_n(N_{\gamma_1}, \ldots, N_{\gamma_{P_\gamma}})} \sum_{k=1}^{P_\gamma} \mu_{\gamma, \gamma k}(x_k). \quad (C.33)
$$

Clearly $L^* \in \Lambda_\gamma(n)$. We then have from (3.11) and (3.12)

$$
\mu_\gamma(n) + \frac{P_\gamma - 1}{\text{var}(V_\gamma)} \geq \mathcal{E}^{-1}(V_\gamma | L^*) + \frac{P_\gamma - 1}{\text{var}(V_\gamma)} \\
= \sum_{k=1}^{P_\gamma} \mathcal{E}^{-1}(V_\gamma | L_{\gamma k}^*) \\
= \sum_{k=1}^{P_\gamma} \mu_{\gamma, \gamma k}(x_k^*) \\
= \max_{x \in \Delta_n(N_{\gamma_1}, \ldots, N_{\gamma_{P_\gamma}})} \sum_{k=1}^{P_\gamma} \mu_{\gamma, \gamma k}(x_k). \quad (C.34)
$$

Thus from (C.33) and (C.34) we have

$$
\mu_\gamma(n) = \mathcal{E}^{-1}(V_\gamma | L^*) = \max_{x \in \Delta_n(N_{\gamma_1}, \ldots, N_{\gamma_{P_\gamma}})} \sum_{k=1}^{P_\gamma} \mu_{\gamma, \gamma k}(x_k) - \frac{P_\gamma - 1}{\text{var}(V_\gamma)}, \quad (C.35)
$$

which proves Claim 44.

**Claim 45** If $L \in \mathcal{L}_{\gamma k}(n)$ then $L \in \mathcal{L}_{\gamma, \gamma k}(n)$ and vice versa.

Denote an arbitrary leaf node of the tree of $\gamma \backslash k$ as $C$. Then $V_\gamma, V_{\gamma k},$ and $C$ are related through

$$
V_{\gamma k} = q_{\gamma k} V_{\gamma} + W_{\gamma k}, \quad (C.36)
$$
and

\[ C = \eta V_k + F \]  \hspace{1cm} (C.37)

where \( \eta \) and \( \varrho_{\gamma k} \) are scalars and \( W_{\gamma k}, F \) and \( V_\gamma \) are independent random variables. We note that by definition \( \text{var}(V_\gamma) > 0 \) \( \forall \gamma \) (see Definition 15). From Lemma 42 we have

\[
\frac{\text{cov}(V_{\gamma k}, C)}{\text{cov}(V_\gamma, C)} = \left( \frac{\text{var}(V_{\gamma k})}{\text{var}(V_\gamma)} \right)^{1/2} \left( \frac{\varrho_{\gamma k}^2 + \text{var}(W_{\gamma k})}{\varrho_{\gamma k}^2} \right)^{1/2}
\]

\[ =: \xi_{\gamma,k} \]

\[ \geq \left( \frac{\text{var}(V_{\gamma k})}{\text{var}(V_\gamma)} \right)^{1/2}. \]  \hspace{1cm} (C.38)

From (C.38) we see that \( \xi_{\gamma,k} \) is not a function of \( C \).

Denote the covariance between \( V_\gamma \) and leaf node vector \( L = [l_i] \in \Lambda_{\gamma k}(n) \) as \( \Theta_{\gamma,L} = [\text{cov}(V_\gamma, l_i)]^T \). Then (C.38) gives

\[ \Theta_{\gamma,k,L} = \xi_{\gamma,k} \Theta_{\gamma,L}. \]  \hspace{1cm} (C.39)

From (3.17) we have

\[ E(V_\gamma | L) = \text{var}(V_\gamma) - \varphi(\gamma, L) \]  \hspace{1cm} (C.40)

where \( \varphi(\gamma, L) = \Theta_{\gamma,L}^T S_{L}^{-1} \Theta_{\gamma,L} \). Note that \( \varphi(\gamma, L) \geq 0 \) since \( S_{L}^{-1} \) is positive semi-definite. Using (C.39) we similarly get

\[ E(V_{\gamma k} | L) = \text{var}(V_{\gamma k}) - \frac{\varphi(\gamma, L)}{\xi_{\gamma,k}^2}. \]  \hspace{1cm} (C.41)

From (C.40) and (C.41) we see that \( E(V_\gamma | L) \) and \( E(V_{\gamma k} | L) \) are both minimized over \( L \in \Lambda_{\gamma k}(n) \) by the same leaf vector that maximizes \( \varphi(\gamma, L) \). This proves Claim 45.

**Claim 46** \( \mu_{\gamma,k}(n) \) is a positive, non-decreasing, and concave function of \( n \), \( \forall k, \gamma \).

We start at a node \( \gamma \) at one scale from the bottom of the tree and then move up the tree.

**Initial Condition:** Note that \( V_{\gamma k} \) is a leaf node. From (3.2) and (C.40) we obtain

\[ E(V_\gamma | V_{\gamma k}) = \text{var}(V_\gamma) - \frac{(\varrho_{\gamma k} \text{var}(V_\gamma))^2}{\text{var}(V_{\gamma k})} \leq \text{var}(V_\gamma). \]  \hspace{1cm} (C.42)
For our choice of $\gamma$, $\mu_{\gamma,k}(1)$ corresponds to $E^{-1}(V_{\gamma}|V_{\gamma,k})$ and $\mu_{\gamma,k}(0)$ corresponds to $1/\text{var}(V_{\gamma})$. Thus from (C.42), $\mu_{\gamma,k}(n)$ is positive, non-decreasing, and concave (trivially since $n$ takes only two values here).

**Induction Step:** Given that $\mu_{\gamma,k}(n)$ is a positive, non-decreasing, and concave function of $n$ for $k = 1, \ldots, P_{\gamma}$, we prove the same when $\gamma$ is replaced by $\gamma \uparrow$. Without loss of generality choose $k$ such that $(\gamma \uparrow)k = \gamma$. From (3.12), (3.14), (C.40), (C.41) and Claim 45, we have for $L \in \mathcal{L}_{\gamma}(n)$

$$\mu_{\gamma}(n) = \frac{1}{\text{var}(V_{\gamma})} \cdot \frac{1}{1 - \frac{\var(\gamma,L)}{\text{var}(V_{\gamma})}} \quad \text{(C.43)}$$

and

$$\mu_{\gamma \uparrow,k}(n) = \frac{1}{\text{var}(V_{\gamma \uparrow})} \cdot \frac{1}{1 - \frac{\var(\gamma,L)}{\mathcal{L}_{\gamma \uparrow,k} \text{var}(V_{\gamma \uparrow})}} \quad \text{(C.44)}$$

From (C.35), the assumption that $\mu_{\gamma,k}(n) \forall k$ is a positive, non-decreasing, and concave function of $n$, and Lemma 18 we have that $\mu_{\gamma}(n)$ is a non-decreasing and concave function of $n$. Note that by definition (see (3.12)) $\mu_{\gamma}(n)$ is positive. This combined with (3.2), (C.43), (C.44), (C.38) and Lemma 43, then prove that $\mu_{\gamma \uparrow,k}(n)$ is also positive, non-decreasing, and concave. \qed
Appendix D

Proofs for covariance trees

We now prove a lemma which we then use to prove Theorem 28.

As a first step we compute the leaf arrangements $L$ which maximize and minimize the sum of all elements of $S_L = [s_{i,j}(L)]$. We restrict our analysis to a covariance tree with depth $N$ and in which each node (excluding leaf nodes) has $\sigma$ child nodes. We introduce some notation. Define

$$\Gamma^{(u)}(p) := \{L : L \in \Lambda_\emptyset(\sigma^p) \text{ and } L \text{ is a uniform leaf node set}\} \quad \text{(D.1)}$$

and

$$\Gamma^{(c)}(p) := \{L : L \text{ is a clustered leaf set of a node at scale } N - p\} \quad \text{(D.2)}$$

for $p = 0, 1, \ldots, N$. We number nodes at scale $m$ in an arbitrary order from $q = 0, 1, \ldots, \sigma^m - 1$ and refer to a node by the pair $(m, q)$.

**Lemma 47** Assume a positive correlation progression. Then, $\sum_{i,j} s_{i,j}(L)$ is minimized over $L \in \Lambda_\emptyset(\sigma^p)$ by every $L \in \Gamma^{(u)}(p)$ and maximized by every $L \in \Gamma^{(c)}(p)$. For a negative correlation progression, $\sum_{i,j} s_{i,j}(L)$ is maximized by every $L \in \Gamma^{(u)}(p)$ and minimized by every $L \in \Gamma^{(c)}(p)$.

**Proof:** Set $p$ to be an arbitrary element in $\{1, \ldots, N - 1\}$. The case of $p = 0$ and $p = N$ is trivial. Let $\vartheta_m = \#\{s_{i,j}(L) \in S_L : s_{i,j}(L) = c_m\}$ be the number of elements of $S_L$
equal to \( c_m \). Define \( a_m := \sum_{k=0}^m \vartheta_k, m \geq 0 \) and set \( a_{-1} = 0 \). Then

\[
\sum_{i,j} s_{i,j} = \sum_{m=0}^N c_m \vartheta_m = \sum_{m=0}^{N-1} c_m (a_m - a_{m-1}) + c_N \vartheta_N \\
= \sum_{m=0}^{N-1} c_m a_m - \sum_{m=-1}^{N-2} c_{m+1} a_m + c_N \vartheta_N \\
= \sum_{m=0}^{N-2} (c_m - c_{m+1}) a_m + c_{N-1} a_{N-1} - c_0 a_{-1} \\
+ c_N \vartheta_N \\
= \sum_{m=0}^{N-2} (c_m - c_{m+1}) a_m + \text{constant}, \tag{D.3}
\]

where we used the fact that \( a_{N-1} = a_N - \vartheta_N \) is a constant independent of the choice of \( L \), since \( \vartheta_N = \sigma^p \) and \( a_N = \sigma^{2p} \).

We now show that \( L \in \Gamma^{(u)}(p) \) maximizes \( a_m, \forall m \) while \( L \in \Gamma^{(c)}(p) \) minimizes \( a_m, \forall m \).

First we prove the results for \( L \in \Gamma^{(u)}(p) \). Note that \( L \) has one element in the tree of every node at scale \( p \).

Case (i) \( m \geq p \). Since every element of \( L \) has distance at most \( p - 1 \) with all other elements, \( a_m = \sigma^p \) which is the maximum value it can take. Case (ii) \( m < p \) (assuming \( p > 0 \)). Let the number of elements of \( L \) belonging to the sub-tree of \( V_{(m+1,q)} \) be \( g_q, q = 0, \ldots, \sigma^{m+1} - 1 \). We have

\[
a_m = \sum_{q=0}^{\sigma^{m+1}-1} g_q (\sigma^p - g_q) = \frac{\sigma^{2p+1+m}}{4} - \sum_{q=0}^{\sigma^{m+1}-1} (g_q - \sigma^p/2)^2 \tag{D.4}
\]
since every element of \( L \) in the tree of \( V_{(m+1,q)} \) must have distance \textit{at most} \( m \) with all nodes \textit{not} in the same tree but must have distance \textit{at least} \( m + 1 \) with all nodes \textit{within} the same tree.

The choice of \( g_q \)'s is constrained to lie on the hyperplane \( \sum_q g_q = \sigma^p \). Obviously the quadratic form of (D.4) is maximized by the point on this hyperplane closest to
the point \((\sigma^p/2, \ldots, \sigma^p/2)\) which is \((\sigma^{p-m-1}, \ldots, \sigma^{p-m-1})\). This is clearly achieved by \(L \in \Gamma^{(u)}(p)\).

Now we prove the results for \(L \in \Gamma^{(c)}(p)\).

Case (i) \(m < N - p\). We have \(a_m = 0\), the smallest value it can take.

Case (ii) \(N - p \leq m < N\). Consider leaf node \(l_i \in L\) which without any loss of generality belongs to the tree of \(V_{(m+1,0)}\). Let \(a_m(l_i)\) be the number of elements of \(L\) to which \(l_i\) has distance less than or equal to \(m\). Now since \(l_i\) has distance less than or equal to \(m\) only with those elements of \(L\) not in the same tree, we must have \(a_m(l_i) \geq \sigma^p - \sigma^{N-m-1}\). Since \(L \in \Gamma^{(c)}(p)\) achieves this lower bound for \(a_m(l_i), \forall i\) and \(a_m = \sum_i a_m(l_i)\), \(L \in \Gamma^{(c)}\) minimizes \(a_m\) in turn. \(\square\)

Let us now study, to what extent the above results transfer to the actual matrix of interest \(S_L^{-1}\). We start with a useful formula.

**Lemma 48** Denote the eigenvalues of \(S_L\) by \(\{f_j; j = 1, \ldots, \sigma^p\}\). Assume that no leaf node of the tree can be expressed as a linear combination of other leaf nodes, implying that \(\{f_j > 0, \forall j\). Set \(D_L = [d_{i,j}]_{\sigma^p \times \sigma^p} := S_L^{-1}\). Then there exist positive numbers \(f_i\) with \(f_1 + \ldots + f_\sigma^p = 1\) such that

\[
\sum_{i,j=1}^{\sigma^p} s_{i,j} = \sigma^p \sum_{j=1}^{\sigma^p} f_j \{f_j\}, \text{ and (D.5)}
\]

\[
\sum_{i,j=1}^{\sigma^p} d_{i,j} = \sigma^p \sum_{j=1}^{\sigma^p} f_j / \{f_j\}. \text{ (D.6)}
\]

Furthermore, for both special cases, \(L \in \Gamma^{(u)}(p)\) and \(L \in \Gamma^{(c)}(p)\), we may choose the weights \(f_j\) such that only one is non-zero.

**Proof:** Since the matrix \(S_L\) is real and symmetric there exists an orthonormal eigenvector matrix \(U = [u_{i,j}]\) that diagonalizes \(S_L\), that is \(S_L = U \Xi U^T\) where \(\Xi\) is diagonal with
eigenvalues $\lambda_j, j = 1, \ldots, \sigma^p$. Define $w_j := \sum_i u_{i,j}$. Then

$$\sum_{i,j} s_{i,j} = 1_{1 \times \sigma^p} S_L 1_{\sigma^p \times 1}$$

$$= (1_{1 \times \sigma^p} U) \Xi (1_{1 \times \sigma^p} U)^T$$

$$= [w_1 \ldots w_{\sigma^p}] \Xi [w_1 \ldots w_{\sigma^p}]^T$$

$$= \sum_j \lambda_j w_j^2.$$  

Further, since $U^T = U^{-1}$ we have

$$\sum_j w_j^2 = (1_{1 \times \sigma^p} U)(U^T 1_{\sigma^p \times 1}) = 1_{1 \times \sigma^p} I_{1_{\sigma^p \times 1}} = \sigma^p.$$  

(D.7)

Setting $f_i = w_i^2 / \sigma^p$ establishes (D.5). Using the decomposition $S_L^{-1} = (U^T)^{-1} \Xi^{-1} U^{-1} = U \Xi^{-1} U^T$ similarly gives (D.6).

Consider the case $L \in \Gamma^{(u)}(p)$.

Since $L = [l_i]$ consists of a symmetrical set of leaf nodes (the set of distances between any element $l_i$ and the rest does not depend on $i$) the sum of the covariances of a leaf node $l_i$ with its fellow leaf nodes does not depend on $i$, and we can set:

$$\lambda^{(u)} := \sum_{j=1}^{\sigma^p} s_{i,j}(L) = c_N + \sum_{m=1}^{\sigma^p-m} c_m.$$  

(D.8)

With the sum of the elements of any row of $S_L$ being identical, the vector $1_{\sigma^p \times 1}$ is an eigenvector of $S_L$ with eigenvalue $\lambda^{(u)}$ equal to (D.8).

Recall that we can always choose a basis of orthogonal eigenvectors which includes $1_{\sigma^p \times 1}$ as the first basis vector. It is well known that the rows of the corresponding basis transformation matrix $U$ will then be exactly these normalized eigenvectors. Since they are orthogonal to $1_{\sigma^p \times 1}$, the sum of their coordinates $w_j (j = 2, \ldots, \sigma^p)$ must be zero. Thus, all $f_i$ but $f_1$ vanish. (The last claim follows also from the observation that the sum of coordinates of the normalized $1_{\sigma^p \times 1}$ equals $w_1 = \sigma^p \sigma^{-p/2} = \sigma^{p/2}$; due to (D.7) $w_j = 0$ for all other $j$.)
Consider the case $L \in \Gamma^{(u)}(p)$.

The reasoning is similar to the above and we can define

$$\lambda^{(c)} := \sum_{j=1}^{p} s_{i,j}(L) = c_N + \sum_{m=1}^{p} \sigma^m c_{N-m}. \quad (D.9)$$

**Proof of Theorem 28**

Due to the special form of the covariance vector $\text{cov}(L, V_\Theta) = \rho 1_{1\times k}$ we observe from (3.17) that minimizing the LMMSE $\mathcal{E}(V_\Theta|L)$ over $L \in \Lambda_\Theta(n)$ is equivalent to maximizing $\sum_{i,j} d_{i,j}(L)$ the sum of the elements of $S_L^{-1}$.

Note that the weights $f_i$ as well as the eigenvalues $\{i\}$ of Lemma 48 depend on the arrangement of the leaf nodes $L$. To avoid confusion, we denote by $\{i\}$ the eigenvalues of $S_L$ for an arbitrary fixed set of leaf nodes $L$, and by $\lambda^{(u)}$ and $\lambda^{(c)}$ the only relevant eigenvalues of $L \in \Gamma^{(u)}(p)$ and $L \in \Gamma^{(c)}(p)$ according to (D.8) and (D.9).

Assume a positive correlation progression, and let $L$ be an arbitrary set of $\sigma^p$ leaf nodes. Lemma 47 and Lemma 48 then imply that

$$\lambda^{(u)} \leq \sum_{j} \lambda_j f_j \leq \lambda^{(c)}. \quad (D.10)$$

Since $S_L$ is positive definite, we must have $\lambda_j > 0$. We may then interpret the middle expression as an expectation of the positive “random variable” $\{i\}$ with discrete law given by $f_i$. Jensen’s inequality applies with the convex function $1/x$ ($x > 0$) and yields

$$\sum_{j} (1/\lambda_j) f_j \geq \frac{1}{\sum_{j} \lambda_j f_j} \geq \frac{1}{\lambda^{(c)}}. \quad (D.11)$$

In other words, $\sum_{i,j} d_{i,j}$ is minimized by $L \in \Gamma^{(c)}(p)$; that is, clustering the nodes in the tree gives the worst LMMSE.

A similar argument holds for the negative correlation progression case which proves the Theorem. \qed
Appendix E

Psuedo-code and computational complexity of water-filling algorithm

structure node{
    var;   /* variance of node */
    M;     /* $\mathcal{N}_\gamma$, where $\gamma$ is this node */
    N;     /* $|L_\gamma|$ where $L$ is the current leaf node set */
    curr_u; /* $\mu_\gamma(N)$ */
    next_u; /* $\mu_\gamma(N+1)$ */
    curr_v; /* $\mu_{\gamma,k}(N)$, where $\gamma k = \gamma$ */
    next_v; /* $\mu_{\gamma,k}(N+1)$ */
    num_child; /* $P_\gamma$ */
    child; /* pointer to location of first child, $\gamma 1$ */
    cov_fac; /* $\xi_{\gamma_1,k}^2$ where $\gamma k = \gamma$ */
    parent; /* pointer to parent, $\gamma \uparrow$ */
    const; /* $\Theta_\gamma$ */
    var_W; /* var($W_\gamma$) */
    max_v_loc; /* child number with highest (next_v-curr_v) */
}

*node create_nodes(){
    /*create an array of ‘node’ structures which sets the parent, child, num_child, var_W, and const fields for all nodes. */
Initialize the root node variance. Return a pointer to the root location.*/

init(node *node){
    node *parent, *child; /*pointers to parent and child of node*/

    parent=node->parent;  /*set parent to parent of current node*/
    child=node->child;    /*set child to first child of current node*/

    node->N=0;

    if (node != root){    /* if the node is not the root */
        node->var=((node->const)² * parent->var) +node->var_W;   
        node->cov_fac=(node->var/parent->var)*[(node->const)²  
            +(node->var_W/parent->var)]/(node->const)²] ;          
        /* set node variance */
        node->curr_v=1/parent->var;
    }/* end if (node!= root) */

    node->curr_u=1/node->var;

    if (node->child != 0){    /* if not a leaf node */
        for k=1 to node->num_child {
            init(child+k-1);      /* initialize all children */
            node->M+=(child+k-1)->M;
        }/*end for loop*/
max_v_loc=find_max_v(node);
    /*find child with maximum (next_v-curr_v) */
node->max_v_loc=max_v_loc;
node->next_v=node->curr_v+(child+max_v_loc-1)->next_v
    -(child+max_v_loc-1)->curr_v;
if (node!=root){
    node->next_v=1/(parent->var - ((1/node->cov_fac)
        *(node->var-(1/node->next_v))));
} ;    /*end if (node != root) */
} ;    /* end if (node->child != 0)*/
else {    /*else if a leaf node*/
    node->M=1;
    node->next_v=1/(parent->var
        -((node->const*parent->var)²/node->var));
    }
return ;
}

int find_max_v(node *node){
    /* find child with maximum (next_v - curr_v) */
int i;
double max=0.0;
int max_v_loc;    /*child number*/
node *child;    /*pointer to child*/

    child=node->child;
for i=1 to node->num_child {
    if (child->next_v - child->curr_v > max){
        max=child->next_v - child->curr_v;
        max_v_loc=i;
        child++; /*move to next child node*/
    } /*end if*/
} /*end for loop*/

/*returns pointer to the next optimal leaf node */
find_next_leaf(node *node){
    node *parent, *child; /*pointers to parent and child*/
    parent=node->parent;
    child=node->child;
    node->N++;

    if (node->num_child==0) { /*if a leaf node */
        best_child=node; /* return this leaf node */
    }
    else{
        best_child=find_next_leaf(child+max_v_loc-1);
        node->curr_u=node->next_u;
        node->curr_v=node->next_v;
        if (node->N < node->M) { /*if nodes remaining in sub-tree*/
            max_v_loc=find_max_v(node);
            /*find child with maximum (next_v-curr_v) */
        }
    }
}

node *parent, *child; /*pointers to parent and child*/
parent=node->parent;
child=node->child;
node->N++;

if (node->num_child==0) { /*if a leaf node */
    best_child=node; /* return this leaf node */
}
else{
    best_child=find_next_leaf(child+max_v_loc-1);
    node->curr_u=node->next_u;
    node->curr_v=node->next_v;
    if (node->N < node->M) { /*if nodes remaining in sub-tree*/
        max_v_loc=find_max_v(node);
        /*find child with maximum (next_v-curr_v) */
    }
node->max_v_loc=max_v_loc;
node->next_u=node->curr_u+(child+max_v_loc-1)->next_v
    -(child+max_v_loc-1)->curr_v;
if (node!=root){
    node->next_v=1/(parent->var - ((1/node->cov_fac)
        *(node->var-(1/node->next_u))));
} /* end if (node ≠ root) */
}; /*end if (node→N < node→M) */
} /*end if (node→num_child=0) */ return(best_child);
}

leaves best_leaves(int n){ /* Determine the best n leaf nodes
to estimate the root*/
    node *root;
    root=create_nodes(); /* create array of node structures */
    init(root); /* initialize the structure array starting
    with root */
    leaves=empty set; /* leaves contains the set of
    leaf indices selected so far*/
    for i=1 to n
        leaves=leavesUfind_next_leaf(root); /* each leaf entry
            consists of a set of child indices of nodes
            connecting the root to the leaf*/
        return(leaves);
    }

**Computational Complexity:** Denote the number of nodes in the tree by $R$ and the
depth of the tree by $N$. The complexity of create_nodes() and init(root) is
$O(R)$ since they create and initialize $R$ structures of type node. The complexity of \texttt{find\_next\_leaf}(root) is $O(N)$ since it updates a fixed amount of information at each node along a path from the root to a leaf along of length $N$. The overall complexity of the water-filling algorithm is thus $O(R + nN)$. If every node in the tree has the same number of children then $N$ is approximately $\log(R)$ thus giving an overall complexity of $O(R + n \log(R))$. 
Appendix F

Pseudo-code for pathChirp algorithm

procedure estimate\_D(q)\{
/* q denotes the vector of a single chirp train’s queuing delays */
for (k = 1 to N - 1) \( E_k = 0; /*initialize*/ \)
i = 1; /* Denotes current packet number */
l = N - 1; /* N=number of chirp packets*/
while(i \( \leq \) N - 1)\{
    if (\( q_i < q_{i+1} \))\{
        j = excursion(q, i, F, L)\;
        choose case(j):
        Case(a): \( j > i \) and \( j \leq N \)
            for (s = i to j - 1)
                if (\( q_s < q_{s+1} \)) \( E_s = R_s; \)
        Case(b): \( j = N + 1 \)
            for (s = i to N - 1) \( E_s = R_i; \)
        l = i;
    \}
\}
/* end choose case */

if \( j = i \) \( j = j + 1; \)

\( i = j; \)

} /* end if */

else

\( i = i + 1; \)

} /* end while*/

\( D = 0; \)

for \( (i = 1 \text{ to } N - 1) \{ /*computing } D*/

if \( (E_i == 0) \)

\( D+ = R_i \Delta_i; /* \text{Case (c)} */ \)

else

\( D+ = E_i \Delta_i; \)

}; /* end of for loop */

\( D = D/\sum_{1 \leq i \leq N-1}(\Delta_i); \)

return \( D; \)

}

/*The excursion segmentation algorithm*/

procedure excursion(q,i,F,L){

\( j = i + 1; \)
max_q = 0;

while((j ≤ N) and (q(j) - q(i) > max_q/F))
{
    max_q = maximum(max_q, q(j) - q(i));
    j = j + 1;
}

if ((j ≥ N)) return j;

if (j - i ≥ L)
    return j;
else
    return i;
Bibliography


   http://people.ee.ethz.ch/~oetiker/webtools/mrtg/.

[4] Network characterization service (NCS),


