

Copyright

by

Han Hee Song

2006

Scalable and Flexible Network Measurement

by

Han Hee Song, B.S.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Arts

The University of Texas at Austin

May 2006

Scalable and Flexible Network Measurement

Approved by
Supervising Committee:

To my family

Acknowledgments

Above all, I would like to acknowledge the guidance I received from my advisors, Dr. Yin Zhang and Dr. Lili Qiu. They have not only given me invaluable advise, but also shown what a good research looks like.

I also give my deep appreciation to Dr. Yan Chen (Northwestern University) for introducing me this fascinating field of network research and guiding my first research work.

I would like to thank my coauthors David Bindel (U.C. Berkeley) and Brian Chavez (Northwestern University) for their help on the research and colleagues at University of Texas.

I thank Korean Ministry of Information and Communication for financial support toward my M.A. degree.

Lastly, I give my special thanks to my father, mother, two brothers and fiancée Myung Kyung Chung for supporting my studies.

HAN HEE SONG

The University of Texas at Austin
May 2006

Scalable and Flexible Network Measurement

Han Hee Song, M.A.

The University of Texas at Austin, 2006

Supervisors: Yin Zhang, Lili Qiu

As the Internet continues to grow, it is increasingly important for users, service providers, and application developers to measure, debug, and understand the performance of the wide-area network. Consequently, network measurement studies have become essential to a wide variety of current and emerging network applications.

In this thesis, we research scalable and flexible network measurement. The thesis consists of two parts: Exact Reconstruction of Network Path Properties (Part I) and Approximate Reconstruction of Network Path Properties (Part II).

In Part I, we solve the problem of selecting a set of paths to monitor in order to reconstruct the *exact* properties of all the paths comprising a network. For an overlay network with N end hosts, existing systems either require $O(N^2)$ measurements, and thus lack scalability, or can only estimate the latency but not

congestion or failures. The algebraic approach we propose in this work selectively monitors k linearly independent paths that can fully describe all the $O(N^2)$ paths. The loss rates and latency of these k paths can be used to estimate the loss rates and latency of all other paths. Through an extensive scalability analysis, we find that for reasonably large N (e.g., 100), the growth of k is bounded to $O(N \log N)$. The findings in this work suggest an upper-bound of the monitoring cost for a scalable network monitoring system.

Once we set up an upper-bound of the monitoring cost this way, in Part II, we present a framework that provides flexibility to a variety of design requirements and better scalability with the cost of little accuracy decrease.

We apply *Bayesian experimental design* to select active measurements that maximize the amount of information we gain about the network path properties subject to given resource constraints. We then apply *network inference* techniques to reconstruct the properties of interest based on the partial, indirect observations we get through these measurements.

By casting network measurement in a general Bayesian decision theoretic framework, we achieve *flexibility*. Our framework can support a variety of design requirements, including (i) differentiated design for providing better resolution to certain parts of the network, (ii) augmented design for conducting additional measurements given existing observations, and (iii) joint design for supporting multiple users who are interested in different parts of the network. Our framework is also *scalable* and can design measurement experiments that span thousands of routers and end hosts.

We develop a toolkit that realizes the framework on PlanetLab. We conduct extensive evaluation using both real traces and synthetic data. The results show that the approach can accurately estimate network-wide and individual path properties by monitoring only within 2-10% of paths. We also demonstrate its effec-

tiveness in providing differentiated monitoring, supporting continuous monitoring, and satisfying the requirements of multiple users.

Contents

Acknowledgments	v
Abstract	vi
List of Tables	xii
List of Figures	xiv
Chapter 1 Introduction	1
1.1 Thesis Overview	2
1.2 Exact Reconstruction of Network Path Properties	2
1.3 Approximate Reconstruction of Network Path Properties	3
1.4 Contribution	4
1.5 Outline	5
Chapter 2 Related Work	6
2.1 Measurement Design	6
2.2 Network Inference	7
Chapter 3 Background	8
3.1 Basic Algorithms	8
3.2 Problem Formulation	10
I Exact Reconstruction of Network Path Properties	12
Chapter 4 Approach	13

4.1	Architecture and Algorithms	13
4.1.1	Algebraic Model	13
4.1.2	Basic Static Algorithms	15
4.2	Scalability Analysis	18
4.3	Dynamic Algorithms for Topology Changes	20
4.3.1	Path Additions and Deletions	20
4.3.2	End Hosts Join/Leave the Overlay	22
4.3.3	Routing Changes	22
4.4	Load Balancing and Topology Error Handling	23
4.4.1	Measurement Load Balancing	23
4.4.2	Handling Topology Measurement Errors	23
Chapter 5 Evaluation		25
5.1	Evaluation	25
5.1.1	Metrics	25
5.1.2	Simulation Methodology	26
5.1.3	Results for Different Topologies	27
5.1.4	Results for Different Link Loss Rate Distribution and Running Time	32
5.1.5	Results for Measurement Load Balancing	33
5.1.6	Results for Topology Changes	33
5.2	Internet Experiments	35
5.2.1	Methodology	35
5.2.2	Results	38
Chapter 6 Summary		41
II Approximate Reconstruction of Network Path Properties		42
Chapter 7 Approach		43
7.1	Design of Experiments	43
7.1.1	Bayesian Experimental Design	44
7.1.2	Flexibility	50
7.1.3	Non-Bayesian Designs	51

7.2	Inference Algorithms	53
7.2.1	L_2 Norm Minimization	54
7.2.2	L_1 Norm Minimization	55
7.2.3	Maximum Entropy Estimation	55
7.3	Toolkit Development	56
Chapter 8 Evaluation		58
8.1	Evaluation Methodology	58
8.1.1	Accuracy Metric	58
8.1.2	Dataset Description	58
8.1.3	Experimental Parameters	60
8.2	Evaluation Results	61
8.2.1	Basic Framework	61
8.2.2	Flexibility of Measurement Design	68
8.2.3	Effects of Prior Information	72
Chapter 9 Summary		74
Chapter 10 Conclusion and Future Work		75
10.1	Conclusion	75
10.2	Future Work	76
Bibliography		77
Vita		83

List of Tables

3.1	Table of notations	8
4.1	Table of notations in addition to Table 3.1	13
5.1	Simulation results for three types of BRITE router topologies: Barabasi-Albert (top), Waxman (middle) and hierarchical model (bottom). OL gives the number of end hosts on the overlay network. AP shows the number of links after pruning (<i>i.e.</i> , remove the nodes and links that are not on the overlay paths). MPR (monitored path ratio) is the fraction of the total end-to-end paths which we monitor. FP is the false positive rate.	30
5.2	Simulation results for a real router topology of 284,805 nodes. MPR and FP are defined the same as in Table 5.1.	31
5.3	Measurement load (as sender or receiver) distribution for various BRITE topologies. OL Size is the number of end hosts on overlay. “LB” means with load balancing, and “NLB” means without load balancing.	31
5.4	Simulation results with model $LLRD_2$. Use the same Barabasi-Albert topologies as in Table 5.1. Refer to Table 5.1 for statistics like rank. FP is the false positive rate. OL means overlay network.	32
5.5	Simulation results for adding end hosts on a real router topology. FP is the false positive rate. Denoted as “+added_value (total_value)”.	34
5.6	Simulation results for deleting end hosts on a real router topology. FP is the false positive rate. Denoted as “-reduced_value (total_value)”.	35
5.7	Simulation results for removing a link from a real router topology.	35

5.8	Distribution of selected PlanetLab hosts.	36
5.9	Loss rate distribution: lossy vs. non-lossy and the sub-percentage of lossy paths.	37
7.1	Inference algorithms. MinL2 and MinL1 can optionally incorporate the nonnegativity constraints: $\mathbf{x} \geq 0$, resulting in MinL2_nonNeg and MinL1_nonNeg , respectively.	54
8.1	Summary of PlanetLab traces used for evaluation.	59
8.2	Summary of synthetic data used for evaluation.	59

List of Figures

4.1	Matrix size representations.	14
4.2	Sample overlay network.	14
4.3	Path (row) selection algorithm	16
4.4	Regression of k in various functions of n under different router-level topologies. Top: Barabasi-Albert model of 20K nodes (left), and Waxman model of 10K nodes (right). Bottom: hierarchical model (AS-level: Barabasi-Albert and router level: Waxman) of 20K nodes (left) and a real topology of 284K routers (right).	17
4.5	Path deletion algorithm	21
5.1	Cumulative distribution of absolute errors (left) and error factors (right) under Gilbert loss model for various topologies.	29
5.2	Histogram of the measurement load distribution (as sender) for an overlay of 300 end hosts on a 5000-node Barabasi-Albert topology.	33
5.3	Sensitivity test of sending frequency	37
5.4	Cumulative percentage of the coverage and the false positive rates for lossy path inference in the 100 experiments.	38
5.5	Cumulative percentage of the absolute errors and error factors for the experiment with the worst accuracy in coverage.	39
5.6	Cumulative percentage of the 95 percentile of absolute errors and error factors for the 100 experiments.	40
7.1	Sequential path selection for Bayesian designs.	48
7.2	SVD based path selection algorithm	52
7.3	QR based path selection algorithm	53

7.4	Our toolkit architecture.	57
8.1	Comparison of inference algorithms for delay estimation in PlanetLab-RTT using A-optimal design.	61
8.2	Comparison of inference algorithms for loss estimation in PlanetLab-loss using A-optimal design.	62
8.3	Comparison of experimental designs for estimating network-wide delay using MinL2 inference algorithm.	66
8.4	Comparison of experimental designs for estimating network-wide delay using MinL1_nonNeg inference algorithm.	66
8.5	Comparison of experimental designs for per-path delay inference using MinL1_nonNeg inference algorithm.	67
8.6	Comparison of experimental designs for per-path loss inference using MinL1_nonNeg inference algorithm (simulation uses the Gilbert loss model).	67
8.7	Use differentiated design based on A-optimal design to provide a higher resolution in estimating a selected set of preferred paths in PlanetLab-RTT.	68
8.8	Comparison of various augmented design schemes in PlanetLab-RTT.	69
8.9	Comparison of different design modes in handling multi-user scenarios using PlanetLab-RTT, where all modes use the A-optimal design.	70
8.10	Comparison of different design schemes using their best modes on PlanetLab-RTT.	72
8.11	Effects of the enhanced prior on PlanetLab-RTT.	73

Chapter 1

Introduction

The explosive growth of the Internet has facilitated the emerge of various network applications ranging from the simple and wide spread World Wide Web (WWW) to a more innovative and intricate technologies such as ISP performance management, traffic engineering, content distribution, overlay routing, and peer-to-peer applications. However, the Internet being an enormous, heterogeneous, and constantly evolving infrastructure makes it difficult to fit new services that requires a good knowledge of their surroundings.

Network measurement is essential to a wide variety of these existing and emerging network applications. For example, ISPs and enterprise networks put increased focus on network performance, and demand capabilities for detailed performance measurement in networks of hundreds or even thousands of nodes. Performance monitoring also becomes a critical capability that allows overlays and peer-to-peer networks to detect and react to changing network conditions.

While much progress has been made in network measurement, two significant challenges remain. First, large-scale network management applications often require the ability to efficiently monitor the whole network. The quadratic growth in the number of network paths with respect to the number of network nodes makes it impractical to measure every path. Second, existing techniques are often tailored to specific application needs, and thus lack the flexibility to accommodate applications with different requirements.

1.1 Thesis Overview

We address these challenges in the following two steps. First, in Exact Reconstruction of Network Path Properties (Part I), we devise a network monitoring system that only requires a small subset of paths in order to rebuild the *exact* properties of all the paths in the network. For an overlay network of N end hosts, existing *general metric* systems [49] require $O(N^2)$ measurements. However, the linear algebraic approach we develop bounds the number of measurements we need to $O(N \log N)$.

Once we set up an upper-bound of the monitoring cost this way, in Approximate Reconstruction of Network Path Properties (Part II), we present a framework that provides flexibility to a variety of design requirements and better scalability with the cost of little accuracy decrease. The *Bayesian experimental design* and *network inference* techniques we use make it possible to build a unified framework within which a large class of network performance inference problems can be modeled, solved, and evaluated. The framework built on the techniques is flexible to accommodate different design requirements and scalable to conduct experiments that span thousands of routers and end hosts.

1.2 Exact Reconstruction of Network Path Properties

In this work [9], we develop a scalable overlay loss rate monitoring system which provides the best accuracy achievable. Consider an overlay network of N end hosts; we define a path to be a routing path between a pair of end hosts, and a link to be an IP link between routers. A path is a concatenation of links. There are $O(N^2)$ paths among the N end hosts, and we wish to select a minimal subset of paths to monitor so that the loss rates and latencies of all other paths can be inferred.

In our proposed method, we selectively monitor a *basis set* of k paths. Any end-to-end path can be written as a unique linear combination of paths in the basis set. Consequently, by monitoring loss rates for the paths in the basis set, we infer loss rates for all end-to-end paths. This can also be extended to other additive metrics, such as latency. The end-to-end path loss rates can be computed even when the paths contain *unidentifiable links* for which loss rates cannot be computed.

Given the measurements for paths corresponding to the basis set, this linear algebraic scheme can reconstruct the end-to-end performance on all paths *exactly*.

Moreover, we show that the size of the basis set(k) grows as $O(N \log N)$ through linear regression tests on various synthetic and real topologies. We also provide some explanation based on the Internet topology and the AS hierarchy. We then design incremental algorithms for path addition and deletion which only cost $O(k^2)$ time, instead of the $O(N^2 k^2)$ time cost to reinitialize the system. Lastly, to maximize practicality and performance of the measurement, we propose randomized schemes for measurement load balancing and effective schemes to handle topology measurement errors.

In both simulations and PlanetLab experiments, we estimate path loss rates with high accuracy using $O(N \log N)$ measurements. For the PlanetLab experiments, the average absolute error of loss rate estimation is only 0.0027, and the average error factor is 1.1, even though about 10% of the paths have incomplete or nonexistent routing information. The average setup (monitoring path selection) time is 0.75 second, and the online update of the loss rates for all 2550 paths takes only 0.16 second. In addition, we adapt to topology changes within seconds without sacrificing accuracy. The measurement load balancing reduces the load variation and the maximum vs. mean load ratio significantly, by up to a factor of 7.3.

1.3 Approximate Reconstruction of Network Path Properties

To further deal with the problem of scalability and flexibility, we develop NetQuest [53], a flexible measurement framework that can support large-scale continuous network monitoring. NetQuest consists of two key components: *design of experiments* and *network inference*.

We apply Bayesian experimental design to determine the set of active measurements that maximize the amount of information we gain about the network path properties subject to given resource constraints (*e.g.*, probing overhead). Bayesian experimental design is built on solid theoretical foundations, and has found numerous applications in scientific research and practical applications, ranging from software testing to medicine, to biology, and to car crash test. Recognizing its potential, we bring Bayesian experimental design into large-scale network measurement. Making the experimental design applicable to such context involves addressing several challenges. First, it is not clear how to formulate the problem of designing

network measurement under the Bayesian experimental design framework. Second, the traditional Bayesian experimental design often targets at a single application. In our environment, there can be many applications with different design requirements. How to use Bayesian experimental design to support such diverse application requirements is an interesting open problem.

To address the above issues, we first formulate the problem under the Bayesian experimental design framework. We then explore a series of Bayesian design schemes, and use extensive evaluation to identify the design scheme best suited for network monitoring. In addition, we develop techniques to achieve flexibility by designing measurement experiments that maximize the information gain for different design objectives and constraints. In particular, our approach can support the following requirements: (i) *differentiated design* for providing better resolution to certain parts of the network, (ii) *augmented design* for conducting additional measurements given existing observations, and (iii) *joint design* for supporting multiple users interested in different parts of the network.

Based on observations obtained from the measurements, we then use inference techniques to accurately reconstruct the global view of the network without requiring complete information. Our results show that our measurement framework can estimate network-wide average path delay within 15% error by monitoring within 2% paths. It achieves a similar degree of accuracy for estimating individual path properties by monitoring 10% paths. In addition, we demonstrate the flexibility of our measurement framework in providing differentiated monitoring, supporting continuous monitoring, and satisfying the requirements of multiple users.

1.4 Contribution

This thesis makes the following main contributions.

First, for an overlay network with n end hosts, we show that the upper-bound of number of paths to fully describe all the $O(N^2)$ grows as $O(N \log N)$.

Second, by bringing Bayesian experimental design into large-scale network measurement, we maintain reasonably good accuracy with even less number of paths. While Bayesian experimental design has found many applications in other scientific fields, to the best of our knowledge, this is the first time that it is applied to designing active network measurement experiments.

Third, building on top of Bayesian experimental design and inference techniques, we develop a unified framework within which a large class of network performance inference problems can be modeled, solved, and evaluated. Our framework is flexible, and can accommodate different design requirements. Our framework is also scalable, and can design measurement experiments that span thousands of routers and end hosts.

Fourth, we develop a toolkit that implements our framework on Planet-Lab [48]. Using the toolkit we conduct an extensive evaluation of our framework for efficient monitoring of end-to-end network performance. Our results demonstrate the effectiveness and flexibility of our framework.

1.5 Outline

The rest of this thesis is organized as follows. In Chapter 2, we survey related work in large-scale network measurement. We describe the large-scale network measurement problem in Chapter 3. In Part I, we discuss network monitoring system that achieves exact reconstruct of network path properties. And we move further into building a flexible framework for a large-scale network measurement in the subsequent part, Part II. In each of the part, we first present our algorithm design, and show implementation and evaluation, then give brief summarization of the work. Finally, we conclude with a summary of major contributions and future works in Chapter 10.

Chapter 2

Related Work

In this chapter, we summarize related works of two categories: network measurement design (Section 2.1) and network inference (Section 2.2).

2.1 Measurement Design

There are many tomography-based end-to-end network measurement design schemes. Network tomography has been well studied ([14] provides a good survey). Most tomography systems assume limited measurements are available (often in a multicast tree-like structure), and try to infer link characteristics [4, 44] or shared congestion [50] in the middle of the network. However, the problem is under-constrained: there exist *unidentifiable links* [4] with properties that cannot be uniquely determined. In contrast, we are not concerned about the characteristics of *individual* links, and we do not restrict the paths we measure.

Shavitt *et al.* also use algebraic tools to compute distances that are not explicitly measured [52]. Given certain “Tracer” stations deployed and some direct measurements among the Tracers, they search for path or path segments whose loss rates can be inferred from these measurements. Thus their focus is not on Tracer/path selection.

Ozmutlu *et al.* selected a minimal subset of paths to cover all links for monitoring, assuming link-by-link latency is available via end-to-end measurement [43]. But the link-by-link latency obtained from traceroute is often inaccurate. And their approach is not applicable for loss rate because it is difficult to estimate link-by-link

loss rates from end-to-end measurement. A similar approach was taken for selecting paths to measure overlay network [59]. The minimal set cover selected can only give *bounds* for metrics like latency, and there is no guarantee as to how far the bounds are from the real values.

Chua *et al.* [11] developed a statistical framework to monitor network-wide properties. Their technique leverages significant redundancy in network paths to reduce monitoring overhead. As opposed to certain statistical summary, such as average end-to-end path delay in a network, we focus on monitoring the property of every path in a network. We choose to focus on inferring every path property because it gives much more detailed information about the network, and is more directly relevant to a variety of applications. Also statistic summary of network-wide properties can be easily obtained once we have every individual path property.

In Section 7.1.3, we revisit algorithms proposed by us in Part I, by Chua *et al.*, and some other non-Bayesian measurement designs to compare their performance against Bayesian measurement design.

2.2 Network Inference

In the area of network inference, many techniques have been developed in statistical literature to solve underdetermined linear system, such as least-squares methods, entropy maximization, maximum likelihood estimation [2], EM [18], Gibbs sampling [26, 27], shrinkage estimation [23, 35], l_1 -minimization [19, 20, 21], LASSO [61, 62], ridge regression [33], etc. Network researchers have leveraged the statistical techniques to come up with interesting approaches to infer network performance (e.g., [5, 45]), network topology (e.g., [22]), and traffic matrix (e.g., [67]).

Chapter 3

Background

3.1 Basic Algorithms

Symbols	Meanings
P	set of all network paths
S	set of subset of P ($S \subseteq P$)
L	set of links on paths in P
m	number of end-to-end paths ($m = P $)
n	number of IP links ($n = L $)
$A \in \{0, 1\}^{m \times n}$	original routing matrix
$A_S \in \{0, 1\}^{S \times n}$	reduced routing matrix
$\mathbf{x} \in \mathbb{R}^n$	vector of unknown performance on individual links
$\mathbf{y} \in \mathbb{R}^m$	vector of observed end-to-end path performance
$\mathbf{y}_S \in \mathbb{R}^S$	reduced vector of observed path performance
v	vector in $\{0, 1\}^m$ (represents path)

Table 3.1: Table of notations

In this thesis, we focus on monitoring end-to-end performance in large networks. The quantity of interest is a function of the performance on individual links, which may not be directly observable either because those links may belong to a non-cooperative administrative domain, or because full instrumentation of an IP network is considered cost prohibitive. Large-scale network measurement is challenging because the number of paths increases quadratically with the number of nodes, and it is often impractical to probe all the network paths, yet the final quantity of in-

terest may depend on links on all the paths. The goal is then to conduct a small number of active measurements, and infer the quantity of interest based on partial and indirect observations. The problem consists of two key aspects: (i) *design of measurement experiments*, and (ii) *network inference* (also commonly referred to as *network tomography*).

Formally, the problem can be specified as follows.

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \quad (3.1)$$

where \mathbf{x} is the vector of some unknown quantity, \mathbf{y} is the vector of observables, \mathbf{A} is a matrix that associates \mathbf{y} and \mathbf{x} (often referred to as the *routing matrix*). In the context of network performance monitoring, \mathbf{x} is the vector of unknown performance on individual links, \mathbf{y} is the vector of observed performance on a set of end-to-end paths, and the routing matrix $\mathbf{A} = (A_{ij})$ encodes whether link j belongs to path i , *i.e.*,

$$A_{ij} = \begin{cases} 1 & \text{if path } i \text{ contains link } j, \\ 0 & \text{otherwise.} \end{cases} \quad (3.2)$$

Note that our definition of routing matrix applies to both one-way and round-trip performance measurements. For round-trip measurements, the routing matrix can work for asymmetric routes.

The above formulation applies to any additive performance metric, such as delay or $\log\{1 - \text{loss rate}\}$. In the case of representing loss rate to additive metric, we first represent a path by $v \in \{0, 1\}^m$, where the j th entry v_j is one if link j is part of the path, and zero otherwise. Suppose link j drops packets with probability l_j ; then the loss rate p of a path represented by v is given by

$$1 - p = \prod_{j=1}^n (1 - l_j)^{v_j} \quad (3.3)$$

We take logarithms on both sides of (3.3). Then by \mathbf{x} with elements $\mathbf{x}_j = \log(1 - l_j)$, we can rewrite (3.3) as follows:

$$\log(1 - p) = \sum_{j=1}^n \mathbf{y}_j \log(1 - l_j) = \sum_{j=1}^n \mathbf{y}_j \mathbf{x}_j = \mathbf{y}^T \mathbf{x} \quad (3.4)$$

Let p_i be the end-to-end loss rate of the i th path, and let $\mathbf{y} \in \mathbb{R}^m$ be a

column vector with elements $\mathbf{y}_i = \log(1 - p_i)$. Then we can obtain the m equations in form (3.4) as we did in 3.1. Therefore we can represent both delay and loss rate in the same form.

Besides performance estimation, there is another type of tomography problem, commonly referred to as *traffic matrix estimation*, which tries to infer end-to-end traffic demands based on observed link loads. In this context, \mathbf{x} is the vector of unknown traffic demands, \mathbf{y} is the vector of observed link loads. There has been considerable recent progress on traffic matrix estimation [38, 66, 67]. In this thesis, we only consider network performance inference, but the framework and the techniques we develop can also be applied to traffic matrix estimation. We plan to explore this direction in our future research.

3.2 Problem Formulation

Our goal is to estimate $f(\mathbf{x})$, which is a function of link properties \mathbf{x} . One interesting example is $f(\mathbf{x}) = A\mathbf{x}$. In this case, the quantity of interest $f(\mathbf{x})$ represents properties of all network paths. More specifically, when \mathbf{x} is link delay, $f(\mathbf{x})$ is delay on all network paths. Another example is $f(\mathbf{x}) = \frac{1}{m}[1, 1, \dots, 1]_{1 \times m}A\mathbf{x}$, which corresponds to a network-wide average metric (*e.g.*, $f(\mathbf{x})$ is the network-wide average path delay, when \mathbf{x} is link delay).

In large-scale network measurement, it is too expensive to directly measure network properties on all paths. So the goal is to select only a small subset of the paths to probe so that we can still accurately estimate the quantity of interest. We formalize the path selection problem as follows.

Let P be the set of all network paths ($|P| = m$). Let L be the set of links appearing on paths in P ($|L| = n$). The performance on paths in P and the performance on links in L are related according to the linear system $\mathbf{y} = A\mathbf{x}$, where \mathbf{x} is a length- n column vector, \mathbf{y} is a length- m column vector, and A is a $m \times n$ routing matrix.

For any subset $S \subseteq P$ (with $s = |S|$), let A_S be the $s \times n$ sub-matrix of A formed by the s rows corresponding to those paths in S . Similarly, let \mathbf{y}_S be the sub-vector of \mathbf{y} corresponding to the observed performance on those paths in S . The experimental design problem is to select a subset of paths S to probe such that we can estimate $f(\mathbf{x})$ based on the observed performance \mathbf{y}_S , A_S , and thus the

following linear system

$$\mathbf{y}_S = A_S \mathbf{x}, \tag{3.5}$$

In this thesis, we consider the case when $f(\mathbf{x})$ is a linear function

$$f(\mathbf{x}) = F \mathbf{x}, \tag{3.6}$$

where F is a given $r \times n$ matrix.

The other major aspect of network performance monitoring is network inference. Its goal is to infer \mathbf{x} based on A_S and \mathbf{y}_S . The major challenge in network inference is that the linear system $\mathbf{y}_S = A_S \mathbf{x}$ is often under-determined due to partial observations, and can thus have an infinite number of solutions.

Part I

Exact Reconstruction of Network Path Properties

Chapter 4

Approach

4.1 Architecture and Algorithms

4.1.1 Algebraic Model

Symbols	Meanings
N	number of end hosts on the overlay
t	number of identifiable links
$k \leq n$	rank of A
l_i	loss rate on i th link
p_i	loss rate on i th measurement path
\mathbf{x}_i	$\log(1 - l_i)$
\mathbf{y}_i	$\log(1 - p_i)$
p	loss rate along a path
$\mathcal{N}(A)$	null space of A
$\mathcal{R}(A^T)$	row(path) space of A ($== \text{range}(A^T)$)

Table 4.1: Table of notations in addition to Table 3.1

In this section, we introduce an algebraic model of this work. In addition to the notations in Table 3.1, we expand our notations as described in Table 4.1.

Suppose an overlay network spans n IP links. Normally, the number of paths m is much larger than the number of links n (see Fig. 4.1(a)). This suggests that we could select n paths to monitor, use those measurements to compute the link loss rate variables \mathbf{x} , and infer the loss rates of the other paths from (3.1).

However, as we discussed in Section 3.2, A is rank deficient: *i.e.*, $k = \text{rank}(A)$

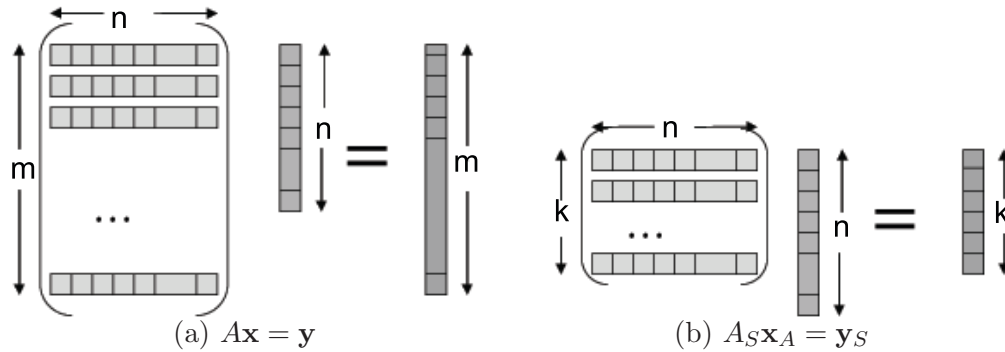


Figure 4.1: Matrix size representations.

and $k < n$. If A is rank deficient, we will be unable to determine the loss rate of some links from (3.1). These links are also called *unidentifiable* in network tomography literature [4].

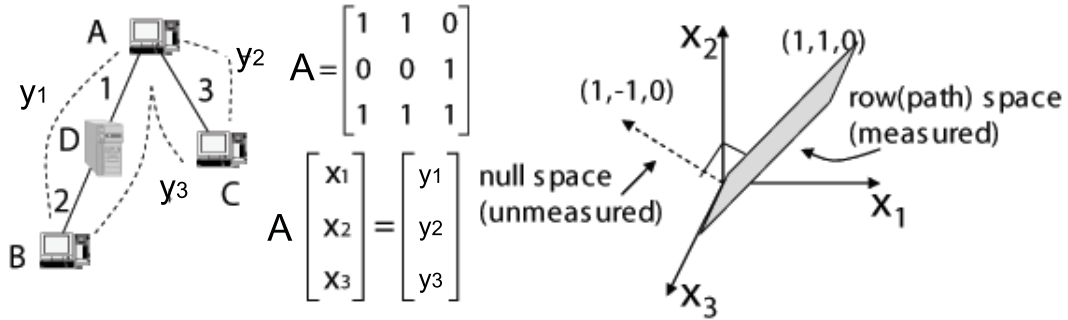


Figure 4.2: Sample overlay network.

Fig. 4.2 illustrates how rank deficiency can occur. There are three end hosts (A, B and C) on the overlay, three links (1, 2 and 3) and three paths between the end hosts. We cannot uniquely solve \mathbf{x}_1 and \mathbf{x}_2 because links 1 and 2 always appear together. We know their sum, but not their difference.

Fig. 4.2 illustrates the geometry of the linear system, with each variable x_i as a dimension. The vectors $\{\alpha [1 \ -1 \ 0]^T\}$ comprise $\mathcal{N}(A)$, the *null space* of A . No information about the loss rates for these vectors is given by (3.1). Meanwhile, there is an orthogonal *row(path) space* of A , $\mathcal{R}(A^T)$, which for this example is a plane $\{\alpha [1 \ 1 \ 0]^T + \beta [0 \ 0 \ 1]^T\}$. Unlike the null space, the loss rate of any vector on the row space can be uniquely determined by (3.1).

To separate the identifiable and unidentifiable components of \mathbf{x} , we decompose \mathbf{x} into $\mathbf{x} = \mathbf{x}_A + \mathbf{x}_N$, where $\mathbf{x}_A \in \mathcal{R}(A^T)$ is its projection on the row space and $\mathbf{x}_N \in \mathcal{N}(A)$ is its projection on the null space (*i.e.*, $A\mathbf{x}_N = 0$). The decomposition of $[\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3]^T$ for the sample overlay is shown below.

$$\mathbf{x}_A = \frac{(\mathbf{x}_1 + \mathbf{x}_2)}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \mathbf{x}_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1/2 \\ \mathbf{y}_1/2 \\ \mathbf{y}_2 \end{bmatrix} \quad (4.1)$$

$$\mathbf{x}_N = \frac{(\mathbf{x}_1 - \mathbf{x}_2)}{2} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \quad (4.2)$$

Thus the vector \mathbf{x}_A can be uniquely identified, and contains all the information we can know from (3.1) and the path measurements. The intuition of our scheme is illustrated through *virtual links* in [8].

Because \mathbf{x}_A lies in the k -dimensional space $\mathcal{R}(A^T)$, only k independent equations of the m equations in (3.1) are needed to uniquely identify \mathbf{x}_A . We measure these k paths to compute \mathbf{x}_A . Since $\mathbf{y} = A\mathbf{x} = A\mathbf{x}_A + A\mathbf{x}_N = A\mathbf{x}_A$, we can compute all elements of \mathbf{y} from \mathbf{x}_A , and thus obtain the loss rate of all other paths. Next, we present more detailed algorithms.

4.1.2 Basic Static Algorithms

The basic algorithms involve two steps. First, we select a basis set of k paths to monitor. Such selection only needs to be done once at setup. Then, based on continuous monitoring of the selected paths, we calculate and update the loss rates of all other paths.

Measurement Paths Selection

To select k linearly independent paths from A , we use standard rank-revealing decomposition techniques [28], and obtain the reduced system (3.5) discussed in Section 3.2. where $A_S \in \mathbb{R}^{k \times n}$ and $\mathbf{y}_S \in \mathbb{R}^k$ consist of k rows of A and \mathbf{y} , respectively. The equation is illustrated in Fig. 4.1(b) (compared with $A\mathbf{x} = \mathbf{y}$).

As shown below, our algorithm is a variant of the QR decomposition with column pivoting [28, p.223]. It incrementally builds a decomposition $A_S^T = QR$,

where $Q \in \mathbb{R}^{n \times k}$ is a matrix with orthonormal columns and $R \in \mathbb{R}^{k \times k}$ is upper triangular.

```

procedure SelectPath(A)
1 for every row(path)  $v$  in  $A$  do
2    $\hat{R}_{12} = R^{-T} A_S v^T = Q^T v^T$ 
3    $\hat{R}_{22} = \|v\|^2 - \|\hat{R}_{12}\|^2$ 
4   if  $\hat{R}_{22} \neq 0$  then
5     Select  $v$  as a measurement path
6     Update  $R = \begin{bmatrix} R & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{bmatrix}$  and  $A_S = \begin{bmatrix} A_S \\ v \end{bmatrix}$ 
   end if
end for

```

Figure 4.3: Path (row) selection algorithm

In general, the A matrix is very sparse; that is, there are only a few nonzeros per row. We leverage this property for speedup. We further use optimized routines from the LAPACK library [1] to implement row selection algorithm so that it inspects several rows at a time. The complexity of row selection algorithm is $O(mk^2)$, and the constant in the bound is modest. The memory cost is roughly $k^2/2$ single-precision floating point numbers for storing the R factor. Notice that the path selection only needs to be executed once for initial setup.

Path Loss Rate Calculations

To compute the path loss rates, we must find a solution to the underdetermined linear system $A_S \mathbf{x}_A = \mathbf{y}_S$. The vector \mathbf{y}_S comes from measurements of the paths. Zhang *et al.* report that path loss rates remain operationally stable in the time scale of an hour [63], so these measurements need not be taken simultaneously.

Given measured values for \mathbf{y}_S , we compute a solution \mathbf{x}_A using the QR decomposition we constructed during measurement path selection [28, 17]. We choose the unique solution \mathbf{x}_A with minimum possible norm by imposing the constraint $\mathbf{x}_A = A_S^T \mathbf{z}$ where $\mathbf{z} = R^{-1} R^{-T} \mathbf{y}_S$. Once we have \mathbf{x}_A , we can compute $\mathbf{y} = A \mathbf{x}_A$, and from there infer the loss rates of the unmeasured paths. The complexity for this step is only $O(k^2)$. Thus we can update loss rate estimates online, as verified in Sec. 5.1.4 and 5.2.2.

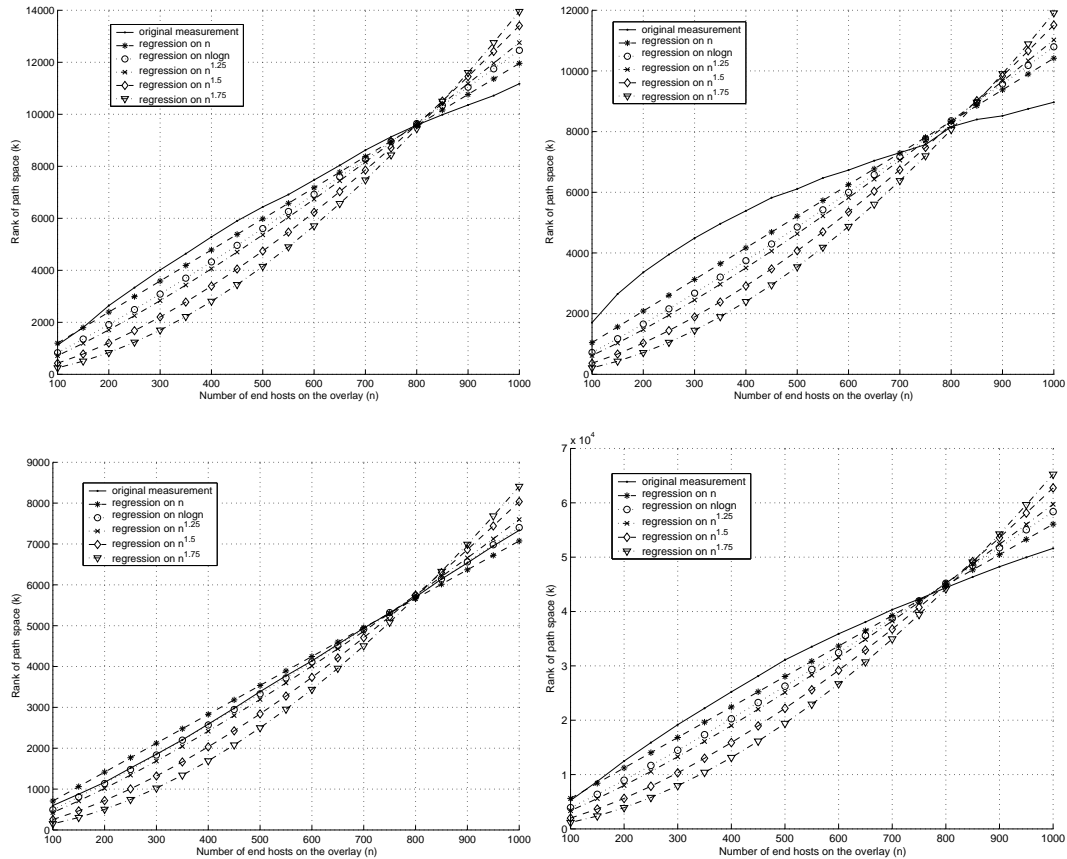


Figure 4.4: Regression of k in various functions of n under different router-level topologies. Top: Barabasi-Albert model of 20K nodes (left), and Waxman model of 10K nodes (right). Bottom: hierarchical model (AS-level: Barabasi-Albert and router level: Waxman) of 20K nodes (left) and a real topology of 284K routers (right).

4.2 Scalability Analysis

An overlay monitoring system is scalable only when the size of the basis set, k , grows relatively slowly as a function of n . Given that the Internet has moderate hierarchical structure [60, 24], we proved that the number of end hosts is no less than half of the total number of nodes in the Internet. Furthermore, we proved that when all the end hosts are on the overlay network, $k = O(N)$ [8].

But what about if only a small fraction of the end hosts are on the overlay? Because A is an m by n matrix, k is bounded by the number of links n . If the Internet topology is a strict hierarchy like a tree, $n = O(N)$, thus $k = O(N)$. But if there is no hierarchy at all (*e.g.* a clique), $k = O(N^2)$ because all the $O(N^2)$ paths are linearly independent. Tangmunarunkit *et al.* found that the power-law degree Internet topology has moderate hierarchy [60]. It is our conjecture that $k = O(N \log n)$.

In this section, we will show through linear regression on both synthetic and real topologies that k is indeed bounded by $O(N \log N)$ for reasonably large N (*e.g.*, 100). We explain it based on the power-law degree distribution of the Internet topology and the AS (Autonomous System) hierarchy.

We experiment with three types of BRITE [37] router-level topologies - Barabasi-Albert, Waxman and hierarchical models - as well as with a real router topology with 284,805 nodes [31]. For hierarchical topologies, BRITE first generates an autonomous system (AS) level topology with a Barabasi-Albert model or a Waxman model. Then for each AS, BRITE generates the router-level topologies with another Barabasi-Albert model or Waxman model. So there are four types of possible topologies. We show one of them as an example because they all have similar trends (see [7] for complete results).

We randomly select end hosts which have the least degree (*i.e.*, leaf nodes) to form an overlay network. We test by linear regression of k on $O(N)$, $O(N \log N)$, $O(N^{1.25})$, $O(N^{1.5})$, and $O(N^{1.75})$. As shown in Fig. 4.4, results for each type of topology are averaged over three runs with different topologies for synthetic ones and with different random sets of end hosts for the real one. We find that for Barabasi-Albert, Waxman and real topologies, $O(N)$ regression has the least residual errors - actually k even grows slower than $O(N)$. The hierarchical models have higher k , and most of them have $O(N \log N)$ as the best fit. Conservatively speaking, we have

$k = O(N \log N)$.

Note that such trend still holds when the end hosts are sparsely distributed in the Internet, *e.g.*, when each end host is in a different access network. One extreme case is the “star” topology - each end host is connected to the same center router via its own access network. In such a topology, there are only N links. Thus $k = O(N)$. Only topologies with very dense connectivity, like a full clique, have $k = O(N^2)$. Those topologies have little link sharing among the end-to-end paths.

The key observation is that when N is sufficiently large, such dense connectivity is very unlikely to exist in the Internet because of the power-law degree distribution. Tangmunarunkit *et al.* found that link usage, as measured by the set of node pairs (source-destination pairs) whose traffic traverses the link, also follows a power-law distribution, *i.e.*, there is a very small number of links that are on the shortest paths of the majority of node pairs. So there is significant amount of link sharing among the paths, especially for backbone links, customer links, and peering links.

Such link sharing can easily lead to rank deficiency of the path matrix for overlay networks. As an example, consider an overlay within a single AS. The AS with the largest number of links (exclusive of customer and peering links) in [54] has 5,300 links. Even considering the coverage factor (55.6% as in Table 2 of [54]), there are at most 9,600 links. Since there are $N(N - 1)$ paths among n nodes, link sharing must occur before $N = 100$; in fact, substantial link sharing is likely to occur for even smaller N .

Now consider an overlay network that spans two ASes connected by c customer/peering links, with $N/2$ nodes in one AS and $N/2$ nodes in the other. The $N^2/2$ cross-AS paths can be modelled as linear combination of $2c \times N + 2c$ virtual links - bi-directional links from each end host to its c peering link routers, and c bi-directional peering links. Thus given c is normally much less than N and can be viewed as a constant, only $O(N)$ paths need to be measured for the $O(N^2)$ cross-AS paths.

Now consider an overlay on multiple ASes. According to [58], there are only 20 ASes (*tier-1* providers) which form the dense core of the Internet. These ASes are connected almost as a clique, while the rest of the ASes have far less dense peering connectivity. So when the size of an overlay is reasonably big (*e.g.*, $N > 100$), the number of customer and peering links that cross-AS paths traverse tends to

grow much slower than $O(N^2)$. For example, a joining end host may only add one customer link to the overlay topology, and share the peering links that have been used by other end hosts. Meanwhile, only a few nodes are needed in a single AS before link sharing occurs in paths within an AS.

We believe this heavy sharing accounts for our empirical observation that $k = O(N)$ in a real router-level topology, and k grows at worst like $O(N \log N)$ in several generated topologies. Note that the real 284,805-router topology represents a fairly complete transit portion of the Internet [31]. In our analysis, we conservatively assume that there is only one end host connecting to each edge router to reduce the possible path sharing, but we still find $k = O(N)$ when $N > 100$.

4.3 Dynamic Algorithms for Topology Changes

During normal operation, new links may appear or disappear, routing paths between end hosts may change, and hosts may enter or exit the overlay network. These changes may cause rows or columns to be added to or removed from A , or entries in A may change. In this section, we design efficient algorithms to incrementally adapt to these changes.

4.3.1 Path Additions and Deletions

The basic building blocks for topology updates are path additions and deletions. We have already handled path additions in path selection algorithm (Figure 4.3); adding a path v during an update is no different than adding a path v during the initial scan of A . In both cases, we decide whether to add v to A_S and update R .

To delete a path that is not in A_S is trivial; we just remove it from A . But to remove a path from A_S is more complicated. We need to update R ; this can be done in $O(k^2)$ time by standard algorithms (see e.g. Algorithm 3.4 in [57, p.338]). In general, we may then need to replace the deleted path with another measurement path. Finding a replacement path, or deciding that no such path is needed, can be done by re-scanning the rows of A as in path selection algorithm (Figure 4.3); however, this would take time $O(mk^2)$.

We now describe path deletion algorithm (Figure 4.5) to delete a path v more efficiently. Suppose v corresponds to the i th row in A_S and the j th row in A , we define $A'_S \in \mathbb{R}^{(k-1) \times n}$ as the measurement path matrix after deleting the i th row,

```

procedure DeletePath( $v, A, A_S, R$ )
1 if deleted path  $v$  is measured then
2    $j =$  index of  $v$  in  $A_S$ 
3    $z = A_S^T R^{-1} R^{-T} e_j$ 
4   Remove  $v$  from  $A$  and  $A_S$ 
5   Update  $R$  (Algorithm 3.4 in [57, p.338])
6    $m = Az$ 
7   if  $\exists i$  such that  $m_i \neq 0$  then
8     Add the  $i$ th path from  $A$  to  $A_S$  (Figure 4.3, steps 2-6)
9   end if
end if
else Remove  $v$  from  $A$ 

```

Figure 4.5: Path deletion algorithm

and $A' \in \mathbb{R}^{(m-1) \times n}$ as the path matrix after removing the j th row. By deleting v from A_S , we reduce the dimension of A_S from k to $k - 1$. Intuitively, our algorithm works in the following two steps.

1. Find a vector z that only describes the direction removed by deleting the i th row of A_S .
2. Test if the path space of A' is orthogonal to that direction, *i.e.*, find whether there is any path $p \in A'$ that has a non-zero component on that direction. If not, no replacement path is needed. Otherwise, replace v with any of such path p , and update the QR decomposition.

Next, we describe how each step is implemented. To find z which is in the path space of A_S but not of A'_S , we solve the linear system $A_S z = e_i$, where e_i is the vector of all zeros except for a one in entry i . This system is similar to the linear system we solved to find \mathbf{x}_A , and one solution is $z = A_S^T R^{-1} R^{-T} e_i$.

Once we have computed z , we compute $m = A'z$, where A' is the updated A matrix. Because we chose z to make $A'_S z = 0$, all the elements of m corresponding to selected rows are zero. Paths such that $m_j \neq 0$ are guaranteed to be independent of A'_S , since if row j of A could be written as $w^T A'_S$ for some w , then m_j would be $w^T A'_S z = 0$. If all elements of m are zero, then z is a null vector for all of A' ; in this case, the dimension k' of the row space of A' is $k - 1$, and we do not need to replace the deleted measurement path. Otherwise, we can find any j such that $m_j \neq 0$ and add the j th path to A'_S to replace the deleted path.

Take the overlay network in Fig. 4.2 for example, suppose A_S is composed of the paths AB and BC , *i.e.*, $A_S = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$. Then we delete path BC , $A'_S = [1 \ 1 \ 0]^T$ and $A' = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Applying path deletion algorithm (Figure 4.5), we have $z = [0 \ 0 \ 1]^T$ and $m = [0 \ 1]^T$. Thus the second path in A' , AC , should be added to A'_S . If we visualize such path deletion in reference to the geometry of the linear system, the path space of A' remains as a plane in Fig. 4.2, but A'_S only has one dimension of the path space left, so we need to add AC to A'_S .

When deleting a path used in A_S , the factor R can be updated in $O(k^2)$ time. To find a replacement row, we need to compute a sparse matrix-vector product involving A , which takes $O(N^2 \times (\text{average path length}))$ operations. Since most routing paths are short, the dominant cost will typically be the update of R . Therefore, the complexity of path deletion algorithm is $O(k^2)$.

4.3.2 End Hosts Join/Leave the Overlay

To add an end host h , we use path selection algorithm (Figure 4.3) to scan all the new paths from h , for a cost of $O(Nk^2)$. However, it is inefficient to delete an end host h by directly using path deletion algorithm (Figure 4.5) to delete all affected paths. If path deletion algorithm is used to delete a path that starts/ends at h , often another path that starts/ends at h is chosen as a replacement – and soon deleted in turn. To avoid this behavior, we remove all these paths from A first, then use the updated A in path deletion algorithm to select replacements as needed during the removal of paths that start/end at h . Each path in A_S can be removed in $O(k^2)$ time; the worst-case total cost of end host deletion is then $O(Nk^2)$.

4.3.3 Routing Changes

In the network, routing changes or link failures can affect multiple paths in A . Previous studies have shown that end-to-end Internet paths generally tend to be stable for significant lengths of time, *e.g.*, for at least a day [47, 65]. So we can incrementally measure the topology to detect changes. Each end host measures the paths to all other end hosts daily, and for each end host, such measurement load can be evenly distributed throughout the day. In addition to the periodic route

measurement, if any path is found to have large loss rate changes, we will check its route instantly.

For each link, we keep a list of the paths that traverse it. If any path is reported as changed for certain link(s), we will examine all other paths that go through those link(s) because it is highly likely that those paths can change their routes as well. We use path selection algorithm and path deletion algorithm to incrementally incorporate each path change.

Unlike $O(N^2)$ approaches (*e.g.*, RON), we need some extra traceroute measurement. However, the key point is that the end-to-end routing remains much more stable than its loss rate, thus requires far less frequent measurement. So the savings on loss rate probing dwarf the traceroute overhead.

4.4 Load Balancing and Topology Error Handling

To further improve the scalability and accuracy, we need to have good load balancing and handle topology measurement errors, as discussed in this section.

4.4.1 Measurement Load Balancing

To avoid overloading any single node or its access link, we evenly distribute the measurements among the end hosts. We randomly reorder the paths in A before scanning them for selection in path selection algorithm (Figure 4.3). Since each path has equal probability of being selected for monitoring, the measurement load on each end host is similar. Note any basis set generated from path selection algorithm is sufficient to describe all paths A . Thus the load balancing has no effect on the loss rate estimation accuracy.

4.4.2 Handling Topology Measurement Errors

As our goal is to estimate the end-to-end path loss rate instead of any interior link loss rate, we can tolerate certain topology measurement inaccuracies, such as incomplete routing information and poor router alias resolution.

For completely untraceable paths, we add a direct link between the source and the destination. In our system, these paths will become selected paths for monitoring. For paths with incomplete routing information, we add links from

where the normal route becomes unavailable (*e.g.*, self loops or displaying “* * *” in traceroute), to where the normal route resumes or to the destination if such anomalies persist until the end. For instance, if the measured route is $(src, ip_1, “* * *”, ip_2, dest)$, the path is composed of three links: (src, ip_1) , (ip_1, ip_2) , and $(ip_2, dest)$. By treating the untraceable path (segment) as a normal link, the resulting topology is equivalent to the one with complete routing information for calculating the path loss rates.

For topologies with router aliases presenting one physical link as several links, we have little need to resolve these aliases. At worst, our failure to recognize the links as the same will result in a few more path measurements because the rank of A will be higher. For these links, their corresponding entries in \mathbf{x}_A will be assigned similar values because they are actually a single link. Thus the path loss rate estimation accuracy is not affected, as verified by Internet experiments in Sec. 5.2. In addition, our system is robust to measurement node failures and node changes by providing bounds on the estimated loss rates.

Chapter 5

Evaluation

5.1 Evaluation

In this section, we present our evaluation metrics, simulation methodology and simulation results.

5.1.1 Metrics

The metrics include path loss rate estimation accuracy, variation of measurement loads among the end hosts, and speed of setup, update, and topology change adaptation.

To compare the inferred loss rate \hat{p} with real loss rate p , we analyze both absolute error and error factor. The absolute error is $|p - \hat{p}|$. We adopt the error factor $F_\varepsilon(p, \hat{p})$ defined in [4] as follows:

$$F_\varepsilon(p, \hat{p}) = \max \left\{ \frac{p(\varepsilon)}{\hat{p}(\varepsilon)}, \frac{\hat{p}(\varepsilon)}{p(\varepsilon)} \right\} \quad (5.1)$$

where $p(\varepsilon) = \max(\varepsilon, p)$ and $\hat{p}(\varepsilon) = \max(\varepsilon, \hat{p})$. Thus, p and \hat{p} are treated as no less than ε , and then the error factor is the maximum ratio, upwards or downwards, by which they differ. We use the default value $\varepsilon = 0.001$ as in [4]. If the estimation is perfect, the error factor is one.

Furthermore, we classify a path to be lossy if its loss rate exceeds 5%, which is the threshold between “tolerable loss” and “serious loss” as defined in [63]. We report the true number of lossy paths, the percentage of real lossy paths identified

(coverage) and the false positive rate, all averaged over five runs of experiment for each configuration.

There are two types of measurement load: 1) sending probes, and 2) receiving probes and computing loss rates. The load reflects the CPU and uplink/downlink bandwidth consumption. For each end host h , its measurement load is linearly proportional to, and thus denoted by the number of monitored paths with h as sender/receiver. Then we compute its variation across end hosts in terms of the *coefficient of variation* (CV) and the *maximum vs. mean ratio* (MMR), for sending load and receiving load separately. The CV of a distribution x , defined as below, is a standard metric for measuring inequality of x , while the MMR checks if there is any single node whose load is significantly higher than the average load.

$$CV(x) = \frac{\text{standard deviation}(x)}{\text{mean}(x)} \quad (5.2)$$

The simulations only consider undirected links, so for each monitored path, we randomly select one end host as sender and the other as receiver. This is applied to all simulations with or without load balancing.

5.1.2 Simulation Methodology

We consider the following dimensions for simulation.

- Topology type: three types of synthetic topologies from BRITE (see Sec. 5.1.3) and a real router-level topology from [31]. All the hierarchical models have similar results, we just use Barabasi-Albert at the AS level and Waxman at the router level as the representative.
- Topology size: the number of nodes ranges from 1000 to 20000¹. Note that the node count includes both internal nodes (i.e., routers) and end hosts.
- Fraction of end hosts on the overlay network: we define end hosts to be the nodes with the least degree. Then we randomly choose from 10% to 50% of end hosts to be on the overlay network. This gives us pessimistic results because other distributions of end hosts will probably have more sharing of

¹20000 is the largest topology we can simulate on a 1.5GHz Pentium 4 machine with 512M memory.

the routing paths among them. We prune the graphs to remove the nodes and links that are not referenced by any path on the overlay network.

- Link loss rate distribution: 90% of the links are classified as “good” and the rest as “bad”. We use two different models for assigning loss rate to links as in [44]. In the first model ($LLRD_1$), the loss rate for good links is selected uniformly at random in the 0-1% range and that for bad links is chosen in the 5-10% range. In the second model ($LLRD_2$), the loss rate ranges for good and bad links are 0-1% and 1-100% respectively. Given space limitations, most results are under model $LLRD_1$ except for Sec. 5.1.4.
- Loss model: After assigning each link a loss rate, we use either a Bernoulli or a Gilbert model to simulate the loss processes at each link. For a Bernoulli model, each packet traversing a link is dropped at independently fixed probability as the loss rate of the link. For a Gilbert model, the link fluctuates between a good state (no packet dropped) and a bad state (all packets dropped). According to Paxson’s observed measurement of Internet [46], the probability of remaining in bad state is set to be 35% as in [44]. Thus, the Gilbert model is more likely to generate bursty losses than the Bernoulli model. The other state transition probabilities are selected so that the average loss rates matches the loss rate assigned to the link.

We repeat our experiments five times for each simulation configuration unless denoted otherwise, where each repetition has a new topology and new loss rate assignments. The path loss rate is simulated based on the transmission of 10000 packets. Using the loss rates of selected paths as input, we compute \mathbf{x}_A , then the loss rates of all other paths.

5.1.3 Results for Different Topologies

For all topologies in Sec. 5.1.2, we achieve high loss rate estimation accuracy. Results for the Bernoulli and the Gilbert models are similar. Since the Gilbert loss model is more realistic, we plot the cumulative distribution functions (CDFs) of absolute errors and error factors with the Gilbert model in Fig. 5.1. For all the configurations, the absolute errors are less than 0.008 and the error factors are less than 1.18. Waxman topologies have similar results, and we omit them in the interest of space.

The lossy path inference results are shown in Table 5.1. Notice that k is much smaller than the number of IP links that the overlay network spans, which means that there are many IP links whose loss rates are unidentifiable. Although different topologies have similar asymptotic regression trend for k as $O(N \log N)$, they have different constants. For an overlay network with given number of end hosts, the more IP links it spans on, the bigger k is. We found that Waxman topologies have the largest k among all synthetic topologies. For all configurations, the lossy path coverage is more than 96% and the false positive ratio is less than 8%. Many of the false positives and false negatives are caused by small estimation errors for paths with loss rates near the 5% threshold.

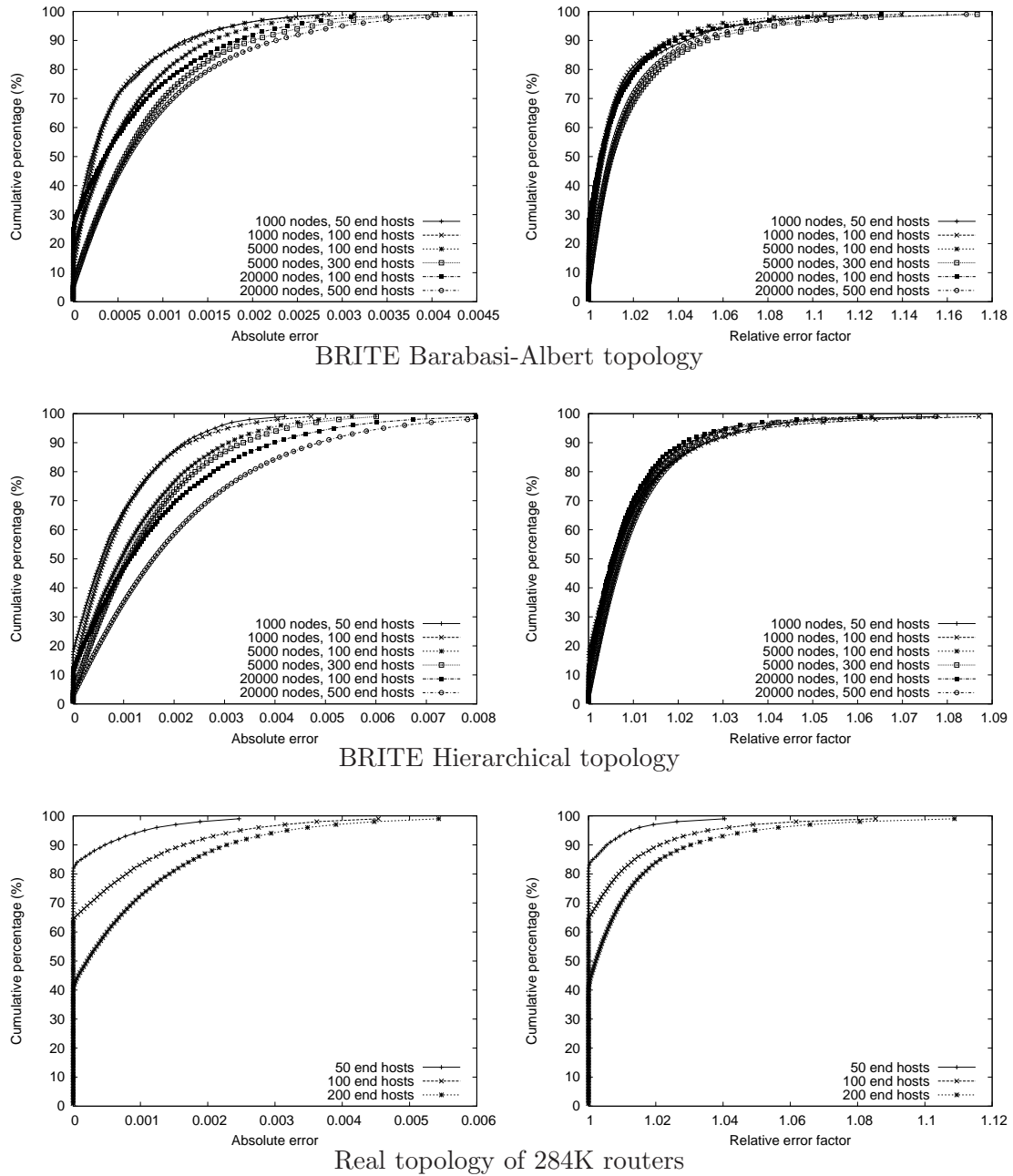


Figure 5.1: Cumulative distribution of absolute errors (left) and error factors (right) under Gilbert loss model for various topologies.

# of nodes	# of end hosts		# of paths(m)	# of links		rank (k)	MPR (k/m)	lossy paths (Bernoulli)			lossy paths (Gilbert)		
	total	OL(N)		original	AP			real	coverage	FP	real	coverage	FP
1000	506	50	1225	1997	443	275	22%	437	99.6%	1.3%	437	100.0%	0.2%
		100	4950		791	543	11%	2073	99.0%	2.0%	1688	99.9%	0.2%
5000	2489	100	4950	9997	1615	929	19%	2271	99.1%	2.0%	2277	99.7%	0.1%
		300	44850		3797	2541	6%	19952	98.6%	4.1%	20009	99.6%	0.3%
20000	10003	100	4950	39997	2613	1318	27%	2738	98.4%	3.4%	2446	99.5%	0.6%
		500	124750		11245	6755	5%	67810	97.8%	5.5%	64733	99.5%	0.4%

# of nodes	# of end hosts		# of paths(m)	# of links		rank (k)	MPR (k/m)	lossy paths (Bernoulli)			lossy paths (Gilbert)		
	total	OL(N)		original	AP			real	coverage	FP	real	coverage	FP
1000	335	50	1225	2000	787	486	40%	704	99.0%	1.1%	579	99.6%	0.4%
		100	4950		1238	909	18%	2544	98.5%	4.6%	2539	99.7%	0.5%
5000	1680	100	4950	10000	2996	1771	36%	3067	97.5%	3.9%	3024	99.5%	0.4%
		300	44850		6263	4563	10%	29135	96.8%	7.1%	28782	99.1%	1.1%
20000	6750	100	4950	40000	5438	2606	53%	3735	98.4%	2.3%	3607	99.6%	0.4%
		500	124750		20621	13769	11%	93049	96.1%	5.7%	92821	99.1%	1.5%

# of nodes	# of end hosts		# of paths(m)	# of links		rank (k)	MPR (k/m)	lossy paths (Bernoulli)			lossy paths (Gilbert)		
	total	OL(N)		original	AP			real	coverage	FP	real	coverage	FP
1000	312	50	1225	2017	441	216	18%	1034	98.8%	2.0%	960	99.6%	0.5%
		100	4950		796	481	10%	4207	98.4%	1.6%	3979	99.6%	0.3%
5000	1608	100	4950	10047	1300	526	11%	4688	99.1%	0.6%	4633	99.8%	0.2%
		300	44850		3076	1787	4%	42331	99.2%	0.8%	42281	99.8%	0.1%
20000	6624	100	4950	40077	2034	613	12%	4847	99.8%	0.2%	4830	100.0%	0.1%
		500	124750		7460	3595	3%	122108	99.5%	0.3%	121935	99.9%	0.1%

Table 5.1: Simulation results for three types of BRITE router topologies: Barabasi-Albert (top), Waxman (middle) and hierarchical model (bottom). OL gives the number of end hosts on the overlay network. AP shows the number of links after pruning (*i.e.*, remove the nodes and links that are not on the overlay paths). MPR (monitored path ratio) is the fraction of the total end-to-end paths which we monitor. FP is the false positive rate.

# of end hosts on overlay (N)	# of paths(m)	# of links after pruning	rank (k)	MPR (k/m)	lossy paths (Bernoulli)			lossy paths (Gilbert)		
					real	coverage	FP	real	coverage	FP
50	1225	2098	1017	83%	891	99.7%	0.9%	912	100.0%	0.2%
100	4950	5413	3193	65%	3570	98.7%	1.9%	3651	99.6%	0.3%
200	19900	12218	8306	42%	14152	97.9%	3.1%	14493	99.6%	0.4%

Table 5.2: Simulation results for a real router topology of 284,805 nodes. MPR and FP are defined the same as in Table 5.1.

# of nodes	OL size (N)	Barabasi-Albert model								hierarchical model							
		CV				MMR				CV				MMR			
		sender		receiver		sender		receiver		sender		receiver		sender		receiver	
		LB	NLB	LB	NLB	LB	NLB	LB	NLB	LB	NLB	LB	NLB	LB	NLB	LB	NLB
1000	50	0.62	1.10	0.56	0.94	2.41	5.91	3.07	4.09	0.52	0.96	0.53	0.87	2.28	4.80	2.51	4.29
	100	0.61	1.42	0.64	1.34	3.21	11.33	3.61	10.67	0.51	1.38	0.47	1.39	2.74	10.06	2.32	10.27
5000	100	0.44	0.89	0.47	0.97	2.25	6.11	2.36	6.50	0.49	1.18	0.53	1.39	2.60	9.18	2.97	10.16
	300	0.52	1.59	0.51	1.51	2.97	18.70	2.74	17.25	0.47	1.72	0.48	1.76	3.47	23.93	4.13	25.76
20000	100	0.36	0.55	0.40	0.59	1.93	3.20	2.29	3.69	0.48	1.17	0.43	1.09	3.04	8.86	2.56	7.09
	500	0.52	1.36	0.53	1.35	2.64	19.21	3.01	16.82	0.46	1.85	0.46	1.89	5.01	25.85	5.56	27.67

Table 5.3: Measurement load (as sender or receiver) distribution for various BRITE topologies. OL Size is the number of end hosts on overlay. “LB” means with load balancing, and “NLB” means without load balancing.

We also test our algorithms in the 284,805-node real router-level topology from [31]. There are 65,801 end host routers and 860,683 links. We get the same trend of results as illustrated in Fig. 5.1 and Table 5.2. The CDFs include all the path estimates, including the monitored paths for which we know the real loss rates. Given the same number of end hosts, the ranks in the real topology are higher than those of the synthetic ones. But as we find in Sec. 4.2, the growth of k is still bounded by $O(N)$.

5.1.4 Results for Different Link Loss Rate Distribution and Running Time

We have also run all the simulations above with model $LLRD_2$. The loss rate estimation is a bit less accurate than it is under $LLRD_1$, but we still find over 95% of the lossy paths with a false positive rate under 10%. Given space limitations, we only show the lossy path inference with the Barabasi-Albert topology model and the Gilbert loss model in Table 5.4.

# of nodes	end hosts		lossy paths (Gilbert)			speed (second)	
	total	OL	real	coverage	FP	setup	update
1000	506	50	495	99.8%	1.1%	0.13	0.08
		100	1989	99.8%	3.0%	0.91	0.17
5000	2489	100	2367	99.6%	3.5%	1.98	0.22
		300	21696	99.2%	1.4%	79.0	1.89
20000	10003	100	2686	98.8%	1.1%	3.00	0.25
		500	67817	99.0%	4.6%	1250	4.33

Table 5.4: Simulation results with model $LLRD_2$. Use the same Barabasi-Albert topologies as in Table 5.1. Refer to Table 5.1 for statistics like rank. FP is the false positive rate. OL means overlay network.

The running time for $LLRD_1$ and $LLRD_2$ are similar, as in Table 5.4. All speed results in this work are based on a 1.5 GHz Pentium 4 machine with 512M memory. Note that it takes about 20 minutes to setup (select the measurement paths) for an overlay of 500 end hosts, but only several seconds for an overlay of size 100. The update (loss rate calculation) time is small for all cases, only 4.3 seconds for 124,750 paths. Thus it is feasible to update online.

5.1.5 Results for Measurement Load Balancing

We examine the measurement load distribution for both synthetic and real topologies, and the results are shown in Table 5.3. Given the space constraints, we only show the results for Barabasi-Albert and hierarchical model. Our load balancing scheme reduces CV and MMR substantially for all cases, and especially for MMR. For instance, a 500-node overlay on a 20000-node network of Barabasi-Albert model has its MMR reduced by 7.3 times.

We further plot the histogram of measurement load distribution by putting the load values of each node into 10 equally spaced bins, and counting the number of nodes in each bin as y -axis. The x -axis denotes the center of each bin, as illustrated in Fig. 5.2. With load balancing, the histogram roughly follow the normal distribution. In contrast, the histogram without load balancing is close to an exponential distribution. Note that the y -axis in this plot is logarithmic: an empty bar means that the bin contains one member, and 0.1 means the bin is empty.

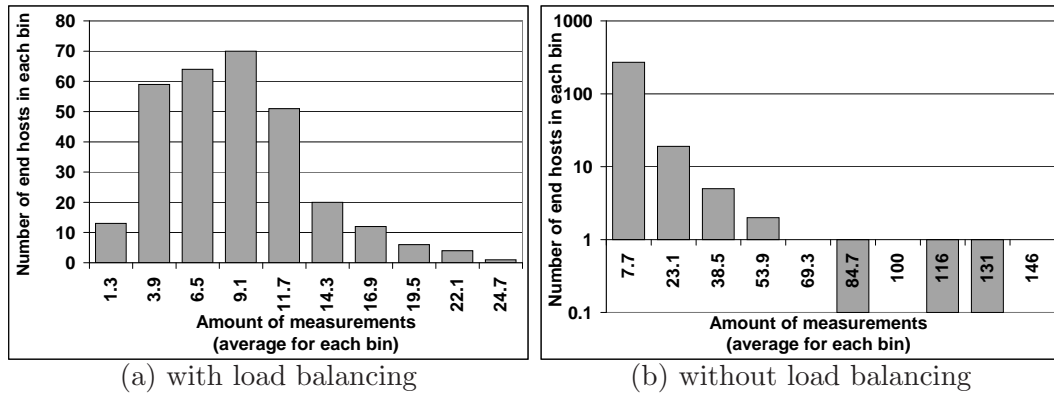


Figure 5.2: Histogram of the measurement load distribution (as sender) for an overlay of 300 end hosts on a 5000-node Barabasi-Albert topology.

5.1.6 Results for Topology Changes

We study two common scenarios in P2P and overlay networks: end hosts joining and leaving as well as routing changes. Again, the Bernoulli and the Gilbert models have similar results, thus we only show those of the Gilbert model.

End hosts join/leave

For the real router topology, we start with an overlay network of 40 random end hosts. Then we randomly add an end host to join the overlay, and repeat the process until the size of the overlay reaches 45 and 50. Averaged over three runs, the results in Table 5.5 show that there is no obvious accuracy degradation caused by accumulated numerical errors. The average running time for adding a path is 125 msec, and for adding a node, 1.18 second. Notice that we add a block of paths together to speedup adding node (Sec. 4.1.2).

Similarly, for removing end hosts, we start with an overlay network of 60 random end hosts, then randomly select an end host to delete from the overlay, and repeat the process until the size of the overlay is reduced to 55 and 50. Again, the accumulated numerical error is negligible as shown in Table 5.6. As shown in Sec. 4.3, deleting a path in A_S is much more complicated than adding a path. With the same machine, the average time for deleting a path is 445 msec, and for deleting a node, 16.9 seconds. We note that the current implementation is not optimized: we can speed up node deletion by processing several paths simultaneously, and we can speed up path addition and deletion with iterative methods such as CGNE or GMRES [3]. Since the time to add/delete a path is $O(k^2)$, and to add/delete a node is $O(Nk^2)$, we expect our updating scheme to be substantially faster than the $O(N^2k^2)$ cost of re-initialization for larger N .

# of end hosts	# of paths	rank	lossy paths		
			real	coverage	FP
40	780	616	470	99.9%	0.2%
+5 (45)	+210 (990)	+221 (837)	+153 (623)	100.0%	0.1%
+5 (50)	+235 (1225)	+160 (997)	+172 (795)	99.8%	0.2%

Table 5.5: Simulation results for adding end hosts on a real router topology. FP is the false positive rate. Denoted as “+added_value (total_value)”.

Routing changes

We form an overlay network with 50 random end hosts on the real router topology. Then we simulate topology changes by randomly choosing a link that is on some

# of end hosts	# of paths	rank	lossy paths		
			real	coverage	FP
60	1770	1397.0	1180.3	99.9%	0.2%
-5 (55)	-285 (1485)	-245.3 (1151.7)	-210.0 (970.3)	99.8%	0.2%
-10 (50)	-260 (1225)	-156.7 (995.0)	-150.6 (819.7)	99.9%	0.1%

Table 5.6: Simulation results for deleting end hosts on a real router topology. FP is the false positive rate. Denoted as “-reduced_value (total_value)”.

path of the overlay and removing of such a link will not cause disconnection for any pair of overlay end hosts. Then we assume that the link is broken, and re-route the affected path(s). Algorithms in Sec. 4.3 incrementally incorporate each path change. Averaged over three runs, results in Table 5.7 show that we adapt quickly, and still have accurate path loss rate estimation.

We also simulate the topology changes by adding a random link on some path(s) of the overlay. The results are similar as above, so we omit them here for brevity.

5.2 Internet Experiments

5.2.1 Methodology

We implemented our system on the PlanetLab [48] testbed, and deployed it on 51 PlanetLab hosts, each from a different organization as shown in Table 5.8. All the international PlanetLab hosts are universities.

First, we measure the topology among these sites by simultaneously running “traceroute” to find the paths from each host to all others. Each host saves its

# of paths affected	40.7
# of monitored paths affected	36.3
# of unique nodes affected	41.7
# of real lossy paths (before/after)	761.0/784.0
coverage (before/after)	99.8%/99.8%
false positive rate (before/after)	0.2%/0.1%
average running time	17.3 seconds

Table 5.7: Simulation results for removing a link from a real router topology.

Areas and Domains		# of hosts	
US (40)	.edu	33	
	.org	3	
	.net	2	
	.gov	1	
	.us	1	
Inter-national (11)	Europe (6)	France	1
		Sweden	1
		Denmark	1
		Germany	1
		UK	2
	Asia (2)	Taiwan	1
		Hong Kong	1
	Canada		2
	Australia		1

Table 5.8: Distribution of selected PlanetLab hosts.

destination IP addresses for sending measurement packets later. Then we measure the loss rates between every pair of hosts. Our measurement consists of 300 trials, each of which lasts 300 msec. During a trial, each host sends a 40-byte UDP packet² to every other host. Usually the hosts will finish sending before the 300 msec trial is finished. For each path, the receiver counts the number of packets received out of 300 to calculate the loss rate.

To prevent any host from receiving too many packets simultaneously, each host sends packets to other hosts in a different random order. Furthermore, any single host uses a different permutation in each trial so that each destination has equal opportunity to be sent later in each trial. This is because when sending packets in a batch, the packets sent later are more likely to be dropped. Such random permutations are pre-generated by each host. To ensure that all hosts in the network take measurements at the same time, we set up sender and receiver daemons, then use a well-connected server to broadcast a “START” command.

Will the probing traffic itself cause losses? We performed sensitivity analysis on sending frequency as shown in Fig. 5.3. All experiments were executed between 1am-3am PDT June 24, 2003, when most networks are free. The traffic rate from or to each host is $(51 - 1) \times \textit{sending_freq} \times 40$ bytes/sec. The number of lossy paths does

²20-byte IP header + 8-byte UDP header + 12-byte data on sequence number and sending time.

not change much when the sending rate varies, except when the sending rate is over 12.8Mbps, since many servers can not sustain that sending rate. We choose a 300 msec sending interval to balance quick loss rate statistics collection with moderate bandwidth consumption.

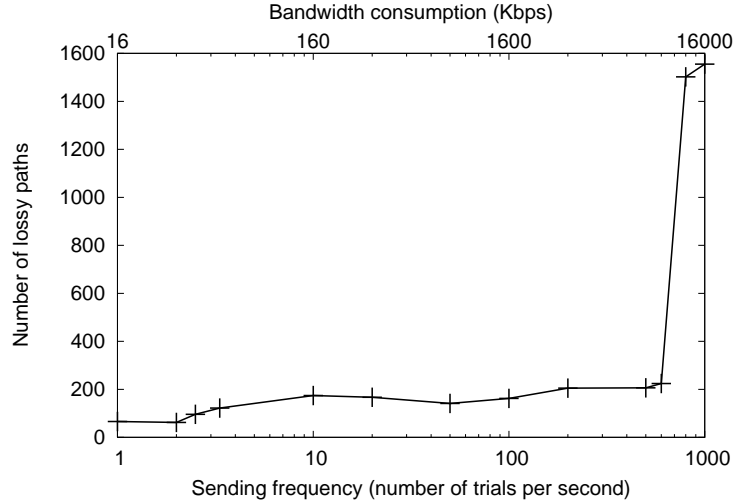


Figure 5.3: Sensitivity test of sending frequency

Note that the experiments above use $O(N^2)$ measurements so that we can compare the real loss rates with our inferred loss rates. In fact, our technique only requires $O(N \log N)$ measurements. Thus, given good load balancing, each host only needs to send to $O(\log N)$ hosts. In fact, we achieve similar CV and MMR for measurement load distribution as in the simulation. Even for an overlay network of 400 end hosts on the 284K-node real topology used before, $k = 18668$. If we reduce the measurement frequency to one trial per second, the traffic consumption for each host is $18668/400 \times 40$ bytes/sec = 14.9Kbps, which is typically less than 5% of the bandwidth of today’s “broadband” Internet links. We can use adaptive measurement techniques in [49] to further reduce the overheads.

loss rate	[0, 0.05]	lossy path [0.05, 1.0] (4.1%)				
		[0.05, 0.1]	[0.1, 0.3]	[0.3, 0.5]	[0.5, 1.0]	1.0
%	95.9%	15.2%	31.0%	23.9%	4.3%	25.6%

Table 5.9: Loss rate distribution: lossy vs. non-lossy and the sub-percentage of lossy paths.

5.2.2 Results

From June 24 to June 27, 2003, we ran the experiments 100 times, mostly during peak hours 9am - 6pm PDT. Each experiment generates $51 \times 50 \times 300 = 765\text{K}$ UDP packets, totaling 76.5M packets for all experiments. We run the loss rate measurements three to four times every hour, and run the pair-wise traceroute every two hours. Across the 100 runs, the average number of selected monitoring paths (A_S) is 871.9, about one third of total number of end-to-end paths, 2550. Table 5.9 shows the loss rate distribution on all the paths of the 100 runs. About 96% of the paths are non-lossy. Among the lossy paths, most of the loss rates are less than 0.5. Though we try to choose stable nodes for experiments, about 25% of the lossy paths have 100% losses and are likely caused by node failures or other reachability problems as discussed in Sec. 5.2.2.

Accuracy and speed

When identifying the lossy paths (loss rates > 0.05), the average coverage is 95.6% and the average false positive rate is 2.75%. Fig. 5.4 shows the CDFs for the coverage and the false positive rate. Notice that 40 runs have 100% coverage and 90 runs have coverage over 85%. 58 runs have no false positives and 90 runs have false positive rates less than 10%.

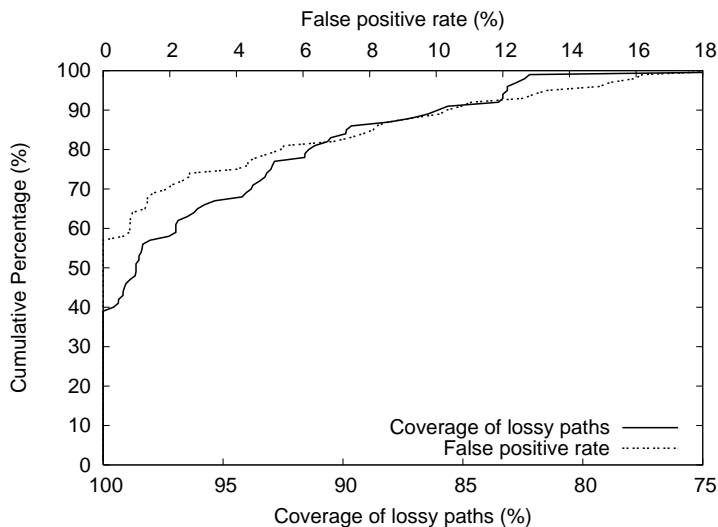


Figure 5.4: Cumulative percentage of the coverage and the false positive rates for lossy path inference in the 100 experiments.

As in the simulations, many of the false positives and false negatives are caused by the 5% threshold boundary effect. The average absolute error across the 100 runs is only 0.0027 for all paths, and 0.0058 for lossy paths. We pick the run with the worst accuracy in coverage (69.2%), and plot the CDFs of absolute errors and error factors in Fig. 5.5. Since we only use 300 packets to measure the loss rate, the loss rate precision granularity is 0.0033, so we use $\varepsilon = 0.005$ for error factor calculation. The average error factor is only 1.1 for all paths.

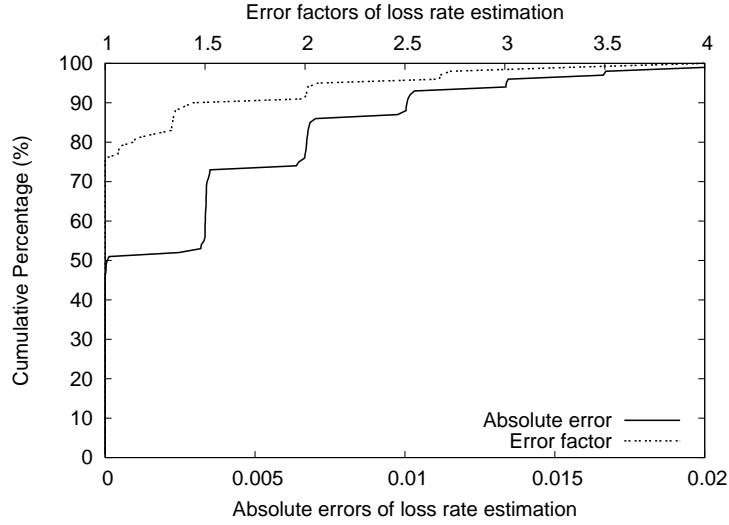


Figure 5.5: Cumulative percentage of the absolute errors and error factors for the experiment with the worst accuracy in coverage.

Even for the worst case, 95% of absolute errors in loss rate estimation are less than 0.014, and 95% of error factors are less than 2.1. To further view the overall statistics, we pick 95 percentile of absolute errors and error factors in each run, and plot the CDFs on those metrics. The results are shown in Fig. 5.6. Notice that 90 runs have the 95 percentile of absolute errors less than 0.0133, and 90 runs have the 95 percentile of error factors less than 2.0.

The average running time for selecting monitoring paths based on topology measurement is 0.75 second, and for loss rate calculation of all 2550 paths is 0.16 second.

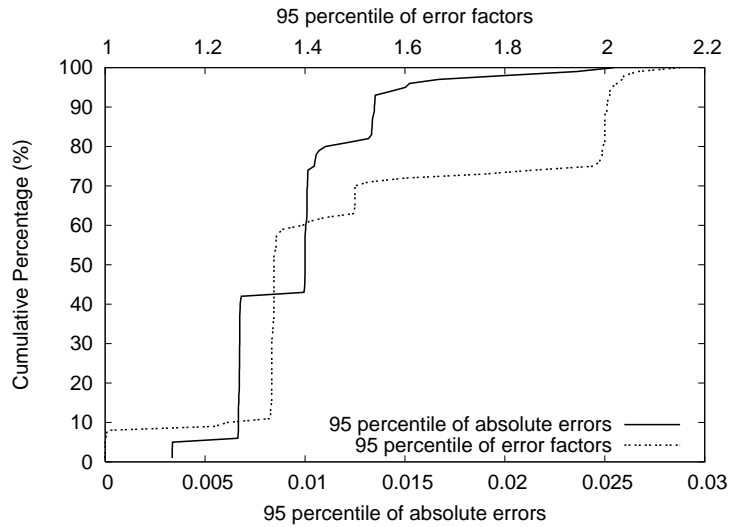


Figure 5.6: Cumulative percentage of the 95 percentile of absolute errors and error factors for the 100 experiments.

Topology error handling

The limitation of traceroute, which we use to measure the topology among the end hosts, led to many topology measurement inaccuracies. As found in [55], many of the routers on the paths among PlanetLab nodes have aliases. We did not use sophisticated techniques to resolve these aliases. Thus, the topology we have is far from accurate. Furthermore, in the PlanetLab experiments, some nodes were down, or were unreachable from certain nodes. Meanwhile, some routers are hidden and we only get partial routing paths. Averaging over 14 sets of traceroutes, 245 out of $51 \times 50 = 2550$ paths have no or incomplete routing information. The accurate loss rate estimation results show that our topology error handling is successful.

Chapter 6

Summary

In this work, for an overlay of N end hosts, we selectively monitor a basis set of $O(N \log N)$ paths which can fully describe all the $O(N^2)$ paths. Then the measurements of the basis set are used to infer the loss rates of all other paths. Our approach works in real time, offers fast adaptation to topology changes, distributes balanced load to end hosts, and handles topology measurement errors. Both simulation and real Internet implementation yield promising results.

Part II

Approximate Reconstruction of Network Path Properties

Chapter 7

Approach

7.1 Design of Experiments

Network measurement, especially when conducted at large scale, requires carefully designed measurement experiments. The design involves specifying all aspects of an experiment and choosing the values of variables that can be controlled before the experiment starts. Making the design decisions is challenging in several ways:

- First, the design space is often quite large, involving a number of control variables. Control variables in network measurement include: choosing the sites to launch experiments from, choosing the subset of paths/links to probe, choosing the type of network characteristics to measure, choosing how to randomize, choosing the granularity, frequency and duration of each experiment, etc. These are all relevant aspects in the design.
- Second, the design is often subject to all kinds of constraints imposed by the network and operations, such as the accessibility of measurement infrastructure, the availability of network resources, and the operational policies and restrictions.
- Third, the design needs to be tailored to accommodate multiple (sometimes conflicting) design objectives. Different users (*e.g.*, VoIP gateway services, versus cable access network providers) often have different notions of performance, and want to monitor different metrics (*e.g.*, delay, loss, or bandwidth).

Given the complexity involved in experimental design, manually making all the design decisions is both time consuming and error prone. It is therefore highly desirable to automate the design process and do so in a mathematically sound manner. Not all aspects of experimental design are amenable to formal mathematical treatment. Choosing the values for the control variables however can be expressed in a coherent mathematical framework through the use of *Bayesian experimental design*, which has gained considerable popularity in the past three decades. Below we first give a brief overview of Bayesian experimental design (see [6, 13] for a detailed review). We then put it into the context of large-scale network measurement and demonstrate how the general framework can be applied to meet common design requirements.

7.1.1 Bayesian Experimental Design

The basic idea in experimental design is that one can improve the statistical inference about the quantities of interest by properly choosing the values of the control variables. This can be formally described in a Bayesian decision theoretic framework as proposed by Lindley in 1972 [36, page 19 and 20]. Below we first set up the framework using decision theoretic terminologies, and then put it into the context of network performance monitoring.

As summarized in [6], Lindley’s framework is the following. Suppose one wants to conduct an experiment on a system with unknown parameters \mathbf{x} drawn from parameter space \mathcal{X} . Before the experiment, a design η must be chosen from some set \mathcal{H} . Through the experiment, data \mathbf{y} from a sample space \mathcal{Y} will be observed. Based on \mathbf{y} a terminal decision d will be chosen from some set \mathcal{D} . In the context of network performance monitoring, the terminal decision d is an estimate of our quantity of interest $f(\mathbf{x})$, where \mathbf{x} is the vector of unknown link performance. A design η refers to a set of paths, S , which we choose to probe. Through the experiment, we observe end-to-end performance on the set of paths in S : \mathbf{y}_S , which satisfies $\mathbf{y}_S = A_S \mathbf{x}$. Here A_S is the routing matrix formed by the set of rows corresponding to paths in S . So the goal is to estimate $f(\mathbf{x})$ based on the observed end-to-end performance on the set of paths in S (*i.e.*, \mathbf{y}_S). So the whole decision process consists of two parts: first the selection of η (*i.e.*, S , in our context), and then the choice of a terminal decision d (*i.e.*, an estimate of $f(\mathbf{x})$, in our context). A general utility function of the

form $U(d, \mathbf{x}, \eta, \mathbf{y})$ is used to reflect the purpose of the experiment. For example, it may reflect the expected accuracy of an estimator for $f(\mathbf{x})$ in the context of network performance inference.

Bayesian experimental design suggests that a good solution to the experimental design problem is the design that maximizes the expected utility of the best terminal decision. More formally, for a given design η , the expected utility of the best terminal decision is

$$U(\eta) = \int_{\mathcal{Y}} \max_{d \in D} \int_{\mathcal{X}} U(d, \mathbf{x}, \eta, \mathbf{y}) p(\mathbf{x}|\mathbf{y}, \eta) p(\mathbf{y}|\eta) d\mathbf{x}d\mathbf{y}, \quad (7.1)$$

where $p(\cdot)$ denotes a probability density function with respect to an appropriate measure. Bayesian experimental design would then choose the design η^* that maximizes the above $U(\eta)$:

$$U(\eta^*) = \max_{\eta \in \mathcal{H}} \int_{\mathcal{Y}} \max_{d \in D} \int_{\mathcal{X}} U(d, \mathbf{x}, \eta, \mathbf{y}) p(\mathbf{x}|\mathbf{y}, \eta) p(\mathbf{y}|\eta) d\mathbf{x}d\mathbf{y}. \quad (7.2)$$

Bayesian Designs for Path Selection

We now apply Bayesian experimental design to solve the path selection problem. Different Bayesian designs can be obtained by choosing different utility functions to assess the quality of individual designs. Below we introduce two such designs: *Bayesian A-optimal design* and *Bayesian D-optimal design*.

Bayesian A-optimal design: For estimating $f(\mathbf{x}) = F\mathbf{x}$, we can use the squared error $\|F\mathbf{x} - F\mathbf{x}_e\|_2^2 = (F\mathbf{x} - F\mathbf{x}_e)^T (F\mathbf{x} - F\mathbf{x}_e)$ to assess the inaccuracy of an estimator $F\mathbf{x}_e$. So a design η can be chosen to maximize the following expected utility

$$U_A(\eta) = - \int (F\mathbf{x} - F\hat{\mathbf{x}})^T (F\mathbf{x} - F\hat{\mathbf{x}}) p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}, \quad (7.3)$$

where $\hat{\mathbf{x}}$ is the estimated \mathbf{x} under the best decision rule d .

To derive an easy-to-use design criterion, we will assume a normal linear system. Specifically, we assume that $\mathbf{y}_S|\mathbf{x}, \sigma^2 \sim A_S\mathbf{x} + N(0, \sigma^2 I)$, where σ^2 is the known variance for the zero mean Gaussian measurement noise, and I is the identity matrix. Suppose the prior information is that $\mathbf{x}|\sigma^2$ is randomly drawn from a multivariate normal distribution with mean vector μ and covariance matrix $\Sigma = \sigma^2 R^{-1}$, where μ and matrix R are known *a priori*.

Let $D(\eta) = (A_S^T A_S + R)^{-1}$. The Bayesian procedure yields

$$U_A(\eta) = -\sigma^2 \text{tr}\{FD(\eta)F^T\}, \quad (7.4)$$

where $\text{tr}\{M\}$ (the trace of a matrix M) is defined as the sum of all the diagonal elements of M .

Maximizing $U_A(\eta)$ thus reduces to minimizing the function

$$\phi_A(\eta) = \text{tr}\{FD(\eta)F^T\}, \quad (7.5)$$

which is commonly referred to as the *Bayesian A-optimality*.

Bayesian D-optimal design: It is also common to replace the quadratic error function in Bayesian A-optimality with an information-theoretic metric. Specifically, one can choose a design that maximizes the expected gain in Shannon information or, equivalently, maximizes the expected Kullback-Leibler distance between the posterior and the prior distributions:

$$\int \log \frac{p(F\mathbf{x}|\mathbf{y}, \eta)}{p(F\mathbf{x})} p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}. \quad (7.6)$$

Since the prior distribution of $F\mathbf{x}$ does not depend on the design η , maximizing (7.6) is equivalent to maximizing

$$U_D(\eta) = \int \log\{p(F\mathbf{x}|\mathbf{y}, \eta)\} p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}. \quad (7.7)$$

Under a normal linear model, the Bayesian procedure yields

$$U_D(\eta) = -\frac{n}{2} \log(2\pi) - \frac{n}{2} - \frac{1}{2} \log \det\{FD(\eta)F^T\}, \quad (7.8)$$

where $\det\{M\}$ denotes the determinant of a matrix M .

Maximizing $U_D(\eta)$ thus reduces to minimizing the function

$$\phi_D(\eta) = \det\{FD(\eta)F^T\}, \quad (7.9)$$

which we define as the *Bayesian D-optimality*.

One problem with (7.9) is that when $\text{rank}(F) < r$ (r is the number of rows in

F), $\det\{FD(\eta)F^T\}$ is always 0 and thus cannot distinguish different designs. Our solution is to redefine

$$\phi_D(\eta) = \Pi_{\text{rank}(F)}\{FD(\eta)F^T\}, \quad (7.10)$$

where $\Pi_k\{M\}$ is the product of the k largest eigenvalues of M .

It is often reasonable to require that a design η remains good under a small perturbation to the function of interest: $f(\mathbf{x}) = F\mathbf{x}$. As a result, if F is not full-rank, we can perturb F slightly to make it full-rank. Therefore, we only need to consider the case when $\text{rank}(F) = \min(r, n)$. In this case, we can simplify (7.10) into

$$\phi_D(\eta) = \begin{cases} \det\{F^T F\} \det\{D(\eta)\} & \text{if } r \geq n, \\ \det\{FD(\eta)F^T\} & \text{otherwise.} \end{cases} \quad (7.11)$$

Note that when $r \geq n$, minimizing $\phi_D(\eta)$ reduces to minimizing $\det\{D(\eta)\}$ or, equivalently, maximizing $\det\{A_S^T A_S + R\}$.

Search Algorithm

Once we have chosen a design criterion $\phi(\eta)$, the next step is to find the optimal design $\eta^* = \arg \min_{\eta} \phi(\eta)$. However, the problem of finding s rows of A to minimize a given design criterion $\phi(\eta)$ is known to be *NP*-complete and it is computationally infeasible to compute the optimal design exactly. To address the issue, we search for a good design using a simple *sequential search algorithm*. The algorithm starts with an empty initial design and then sequentially adds rows to the design. During each iteration, it greedily selects the row that results in the largest reduction in $\phi(\eta)$. The basic algorithm is illustrated in Figure 7.1.

It is possible to further improve the design obtained by the sequential search algorithm using an *exchange algorithm* (e.g., [25, 40, 39, 42]), which tries to reduce $\phi(\eta)$ by iteratively exchanging paths. We have implemented Fedorov's exchange algorithm, which has been shown to yield the best performance among many existing algorithms [42]. However, our experience suggests that the additional path exchanges do not yield noticeable performance improvement in the context of network performance inference. *So we disable the exchange algorithm in the interest of efficiency.*

```

1   $S = \emptyset$  // initialize the path set to be empty
2  for  $iter = 1$  to  $s$  //  $s$  is the desired number of paths
3    for each path  $i \in \{1, 2, \dots, m\} - S$ 
4      // compute the new design criterion after adding path  $i$ 
5       $criteria[i] = \phi(S \cup \{i\})$ 
6    end for
7    // select the path that minimizes the new design criterion
8     $S = S \cup \{\arg \min_i criteria[i]\}$ 
9  end for
10 return  $S$ 

```

Figure 7.1: Sequential path selection for Bayesian designs.

Incremental Update of Design Criterion

For large-scale network measurement, it is essential for the search algorithm to be highly efficient, because the design space is often very large due to the quadratic growth in the number of network paths with respect to the number of network nodes. The major bottleneck of the above search algorithm is computing the new design criterion $\phi(S \cup \{i\})$ after adding a path i (line 5 in Figure 7.1). Recall that both $\phi_A(\eta)$ and $\phi_D(\eta)$ are functions of $FD(\eta)F^T = F(A_S^T A_S + R)^{-1}F^T$. Given the size of $(A_S^T A_S + R)$ and F , it is inefficient to compute $\phi(S \cup \{i\})$ from scratch due to the expensive matrix inversion and matrix multiplication operations involved.

In NetQuest, we significantly improve the efficiency of the search algorithm by applying incremental methods to update the design criterion. With such optimizations, NetQuest has successfully handled routing matrices A with 1,000,000 rows, 50,000 columns, and over 5,000,000 non-zero entries (corresponding to paths among 1,000 nodes). Below we present these incremental update methods in detail.

Notations: Let $S' = S \cup \{i\}$, $M = A_S^T A_S + R$, $M' = A_{S'}^T A_{S'} + R$, $N = FM^{-1}F^T$, $N' = F(M')^{-1}F^T$. With these notations, we have $\phi_A(S) = \text{tr}\{N\}$, $\phi_A(S') = \text{tr}\{N'\}$, $\phi_D(S) = \det\{N\}$, $\phi_D(S') = \det\{N'\}$.

Incremental computation of $\phi_A(S \cup \{i\})$: Let \mathbf{a}_i^T denote the i -th row vector of A . It is easy to verify that $M' = M + \mathbf{a}_i \mathbf{a}_i^T$. That is, M' can be derived from M using a rank-1 matrix update. We have the following result from matrix algebra

$$(M')^{-1} = (M + \mathbf{a}_i \mathbf{a}_i^T)^{-1} = M^{-1} - \alpha \mathbf{u} \mathbf{u}^T, \quad (7.12)$$

where $\mathbf{u} = M^{-1}\mathbf{a}_i$, $\alpha = 1/(1 + \mathbf{a}_i^T \mathbf{u})$.

Combine (7.12) with $N = FM^{-1}F^T$ and $N' = F(M')^{-1}F^T$, we obtain $N' = N - \alpha(F\mathbf{u})(F\mathbf{u})^T$. As a result, we have

$$\text{tr}\{N'\} = \text{tr}\{N\} - \text{tr}\{\alpha(F\mathbf{u})(F\mathbf{u})^T\} = \text{tr}\{N\} - \alpha \|F\mathbf{u}\|_2^2. \quad (7.13)$$

Therefore, we can compute $\phi_A(S \cup \{i\}) = \text{tr}\{N'\}$ incrementally by computing $\mathbf{u} = M^{-1}\mathbf{a}_i$, $\alpha = 1/(1 + \mathbf{a}_i^T \mathbf{u})$, and $\|F\mathbf{u}\|_2^2$ (note that M^{-1} remains fixed for different i). These operations are much more efficient than matrix inversion and matrix multiplication, which are required to compute $\phi_A(S \cup \{i\})$ from scratch.

Incremental computation of $\phi_D(S \cup \{i\})$: Let $\mathbf{b} = \sqrt{\alpha} \cdot F\mathbf{u}$. We have $N' = N - \mathbf{b}\mathbf{b}^T$. Using results in matrix algebra, we have

$$(N')^{-1} = (N - \mathbf{b}\mathbf{b}^T)^{-1} = N^{-1} + \beta\mathbf{v}\mathbf{v}^T, \quad (7.14)$$

$$\det\{N'\} = \det\{N - \mathbf{b}\mathbf{b}^T\} = \frac{1}{\beta} \det\{N\}, \quad (7.15)$$

where $\mathbf{v} = N^{-1}\mathbf{b}$, and $\beta = 1/(1 - \mathbf{b}^T \mathbf{v})$.

Therefore, we can compute $\phi_D(S \cup \{i\}) = \det\{N'\}$ incrementally by computing $\mathbf{v} = N^{-1}\mathbf{b}$, and $\beta = 1/(1 - \mathbf{b}^T \mathbf{v})$ (note that N^{-1} remains fixed for different i). These operations are much more efficient than the matrix inversion and matrix multiplication operations required for computing $\phi_D(S \cup \{i\})$ from scratch.

Further optimization: With the above incremental update methods, we need to update M^{-1} and N^{-1} each time after a new path is added to S (line 8 in Figure 7.1). This takes $O(n^2)$ operations using (7.12) and (7.14). We can further improve efficiency by maintaining the Cholesky factorization of M and N (instead of M^{-1} and N^{-1}), which in general are much sparser and thus more efficient to update incrementally. Note that the only use of M^{-1} and N^{-1} is to compute quantities $\mathbf{u} = M^{-1}\mathbf{a}_i$ and $\mathbf{v} = N^{-1}\mathbf{b}$. We can compute the same quantities using the Cholesky factorization. For example, if we write $M = LL^T$, where L is the lower-triangular factorization of M , we have $\mathbf{u} = (L^T)^{-1}(L^{-1}\mathbf{a}_i)$, which can be computed efficiently using back-substitution without inverting L and L^T .

In NetQuest, we use LDL [16], a MATLAB package highly optimized for incremental Cholesky factorization of sparse matrices. Our experience suggests that the resulting algorithm achieves an order of magnitude speedup over directly up-

dating M^{-1} and N^{-1} .

7.1.2 Flexibility

Our Bayesian experimental design framework is very flexible and can accommodate common requirements in large-scale network measurement. Below we cover three common scenarios.

Differentiated design: In large-scale network performance monitoring, different quantities of interest may not always have the same importance. For example, a subset of paths may belong to a major customer and it is important to monitor those paths more extensively than the other paths. We can accommodate such differentiated design requirement in Bayesian A -optimality by assigning higher weights to those important rows of matrix F in the objective function $f(\mathbf{x}) = F\mathbf{x}$.

Augmented design: Augmented design is useful in large-scale network measurement for several reasons. First, when some of the measurements in a previous design failed, we do not want to design a new experiment completely from scratch, but instead would like to leverage the data that we have already observed as much as possible. Second, augmented design can also be used to design active measurement experiments that complement well with the existing passive measurement (*i.e.*, the additional information gain is maximized). Our design framework can naturally support the augmentation of a previous design. Specifically, let S_0 be the set of rows we obtain in the previous design. We just need to redefine

$$D(\eta) = (A_{S \cup S_0}^T A_{S \cup S_0} + R)^{-1} = (A_S^T A_S + R + A_{S_0}^T A_{S_0})^{-1} \quad (7.16)$$

where $S \cap S_0 = \emptyset$. We can then apply the Bayesian A -optimal and D -optimal design criteria to find S , the set of additional paths to probe. In addition, we also extend QR and SVD, which will be described in Section 7.1.3, to support augmented design by excluding the paths with successful measurements and applying SVD or QR to select a set of paths from the remaining rows.

Multi-user design: In large-scale network performance monitoring, there may be multiple users who are interested in different parts of the network and may have different functions of interest. We can support such scenarios by using a linear combination of individual users' design criteria as the overall design criterion. For-

mally, given a set of users $1, 2, \dots, u$, let $\phi_i(\eta)$ be the preferred design criterion for each user i (which may depend on his/her interest: $f_i(\mathbf{x}) = F_i\mathbf{x}$). We can then use $\phi(\eta) = \sum_i w_i \phi_i(\eta)$ as the combined design criterion, where w_i is the weight associated with user i . As a concrete example, consider the case where Bayesian A -optimality is used by all the users. In this case, we have $\phi_i(\eta) = \text{tr}\{F_i D(\eta) F_i^T\}$. We can show

$$\phi(\eta) = \sum_i w_i \text{tr}\{F_i D(\eta) F_i^T\} = \text{tr}\{F D(\eta) F^T\}, \quad (7.17)$$

where $F = \text{vertcat}\{w_i^{1/2} F_i\}$ is the vertical concatenation of matrices $w_i^{1/2} F_i$. Therefore, the optimal design using the combined design criterion is equivalent to the Bayesian A -optimal design for the combined function: $f(\mathbf{x}) = F\mathbf{x}$. Note that if a subset of users (U) share a common row in their F_i , this row will appear as multiple rows in F . These rows can be merged into a single row with a combined weight of $(\sum_{i \in U} w_i)^{1/2}$. In the special case when $w_i = 1$, the combined weight is simply $|U|^{1/2}$, *i.e.*, the square root of the number of users interested in the row.

Besides the above three common scenarios, our design framework can easily incorporate other constraints in the design space (*e.g.*, the maximum amount of paths that one can probe at each monitoring site, and the number of monitoring sites available). As part of our future work, we plan to further investigate them in the context of specific network monitoring applications.

7.1.3 Non-Bayesian Designs

For performance comparison, we will also examine the following non-Bayesian solutions to the path selection problem.

Rank based solution: In Part I, we presented a linear algebraic approach to efficient monitoring of overlay paths. In the context, given N overlay nodes, there are $N(N - 1)$ different paths. So A has $N(N - 1)$ rows. The quantity of interest is $f(\mathbf{x}) = A\mathbf{x}$ (*i.e.*, the performance on all overlay paths). Given the rank of matrix A , k , the solution is based on the observation that any subset of k independent rows of A , denoted as A_S , are sufficient to span the space of A : $\{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{x} \in \mathbb{R}^n \text{ s.t. } \mathbf{y} = A\mathbf{x}\}$. As a result, given the measurements for paths corresponding to the rows in A_S , one can reconstruct the end-to-end performance on all paths *exactly*. As we have seen from Chapter 6, k gives an upper bound on the number of paths that one

- | | |
|---|--|
| 1 | compute C such that $\Sigma = CC^T$ |
| 2 | compute SVD of AC : $U\nabla V^T = AC$ |
| 3 | extract the first s column vectors of U : $U_s = [\mathbf{u}_1 \mathbf{u}_2 \cdots \mathbf{u}_s]$ |
| 4 | compute the QR factorization with column pivoting on U_s^T :
$QR = (U_s[E, \cdot])^T$ (E is a permutation vector for rows of U_s) |
| 5 | return the first s elements of E : $S = \{e_1, e_2, \dots, e_s\}$ |

Figure 7.2: SVD based path selection algorithm

needs to probe.

SVD based solution: Chua *et al.* [11, 12] presented a statistical framework for monitoring a linear summary of the end-to-end path performance. Their quantity of interest is $f(\mathbf{x}) = \ell^T A\mathbf{x}$, where ℓ is a weight vector. This is a network-wide metric, *e.g.*, average delay of all paths in a network. It is a special case of the inference problem we study in this work.

Chua *et al.* observed that routing matrices encountered in practice generally show significant sharing of links between paths. As a result, A tends to have small *effective* rank compared to their actual matrix rank. That is, a small subset of eigenvalues of $A^T A$ tend to be much larger than the rest. Based on the observation, Chua *et al.* proposed to select a subset of s paths such that the corresponding rows span as much of $\mathcal{R}(A)$ as possible, where $\mathcal{R}(A)$ is the subspace formed by all possible linear combinations of the rows in A . Algorithmically, this problem is equivalent to the subset selection problem in the field of computational linear algebra (see [30, Ch 12]). So [11] adapted the method described in Algorithm 12.2.1 of [30, p. 574], which is based on the singular value decomposition (SVD). Subsequently in [12], Chua *et al.* extends their algorithm to incorporate Σ , the covariance matrix of \mathbf{x} . It assumes that Σ is a diagonal matrix (but allows diagonal elements to be different). The resulting algorithm is summarized in Figure 7.2.

Path selection under general link covariance Σ (*e.g.*, when link performance has spatial dependency) is left as an *open* problem in [12]. Our Bayesian experimental design framework works for any link covariance matrix, and therefore solves this problem.

QR based solution: The third alternative solution is directly based on QR factorization with column pivoting. It is one of the two algorithms proposed by Golub *et al.* [29] for selecting *numerically* independent rows/columns (the other algorithm is the SVD based solution described above). As noted in [29], the QR based solution

- | | |
|---|---|
| 1 | compute C such that $\Sigma = CC^T$ |
| 2 | compute $G = (AC)^T$ |
| 3 | compute the QR factorization with column pivoting on G :
$QR = G[\cdot, E]$ (E is a permutation vector for columns of G) |
| 4 | return the first s elements of E : $S = \{e_1, e_2, \dots, e_s\}$ |

Figure 7.3: QR based path selection algorithm

is generally more efficient than the SVD based solution and often achieves comparable performance. We extend the algorithm in [29] to incorporate the link covariance matrix Σ when Σ is a diagonal matrix. The resulting algorithm is illustrated in Figure 7.3.

Summary: Rank based solution requires monitoring $\text{rank}(A)$ number of paths, which can be expensive in large networks. SVD and QR based solutions can monitor fewer paths at the cost of higher error. We further enhance the flexibility of SVD and QR by extending them to support augmented design. Nevertheless, as we will show, Bayesian experimental design can out-perform the alternatives in the following regards: (i) it achieves higher accuracy when monitoring the same number of paths as SVD and QR, (ii) it is scalable and can be applied to networks of thousands of nodes, and (iii) it is flexible to support diverse design requirements.

7.2 Inference Algorithms

The other major component to the NetQuest framework is network inference, which reconstructs the quantities of interest \mathbf{x} based on partial, indirect observations \mathbf{y} by solving (3.1). Since NetQuest selects only a small number of measurement experiments to conduct, the number of observables can be much smaller than the number of unknowns. Therefore, the linear inverse problem in (3.1) is often *under-determined*. A lot of solutions have been developed for under-determined linear inverse problems. As we noted in [67], many such proposals solve the regularized least-squares problem

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 J(\mathbf{x}), \tag{7.18}$$

where $\|\cdot\|_2$ denotes the L_2 norm, λ is a regularization parameter, and $J(\mathbf{x})$ is a penalization functional. Proposals of this kind have been used in a wide range of fields, with considerable practical and theoretical success when the data match the

Notation	Inference algorithm	Section
MinL2	L_2 norm minimization	§7.2.1
MinL1	L_1 norm minimization	§7.2.2
Entropy	Maximum Entropy Estimation	§7.2.3

Table 7.1: Inference algorithms. **MinL2** and **MinL1** can optionally incorporate the nonnegativity constraints: $\mathbf{x} \geq 0$, resulting in **MinL2_nonNeg** and **MinL1_nonNeg**, respectively.

assumptions of the method, and the regularization functional matches the properties of the estimand. See [32] for a general description of regularization.

In this work, we compare the accuracy of several widely used inference algorithms (summarized in Table 7.1). The goal is to understand how combinations of different experimental design methods and inference algorithms affect the overall inference accuracy.

7.2.1 L_2 Norm Minimization

A common solution to (3.1) is L_2 norm minimization, which corresponds to (7.18) with $J(\mathbf{x}) = \|\mathbf{x}\|_2^2$.

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_2^2. \quad (7.19)$$

If we have prior information that \mathbf{x} is close to an initial solution μ , we can replace $\|\mathbf{x}\|_2$ with $\|\mathbf{x} - \mu\|_2$ in (7.19), resulting in

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x} - \mu\|_2^2. \quad (7.20)$$

(7.20) is also commonly referred to as the Tikhonov regularization [32]. It can be efficiently solved using standard solvers for linear least-squares problems. If desired, one can incorporate the nonnegativity constraints $\mathbf{x} \geq 0$ into (7.20). The resulting nonnegative linear least-squares problem can be solved using PDCO [51], a MATLAB package highly optimized for problems with sparse matrices A .

7.2.2 L_1 Norm Minimization

Another common solution to (3.1) is L_1 norm minimization, which corresponds to (7.18) with $J(\mathbf{x}) = \|\mathbf{x}\|_1$ (i.e., the L_1 norm of \mathbf{x}).

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_1. \quad (7.21)$$

L_1 norm minimization is often used in situations where \mathbf{x} is *sparse*, i.e., \mathbf{x} has only very few large elements and the other elements are all close to 0. This can happen, for instance, in loss inference, where most links have close to 0 loss rate (and thus $\log\{1 - \text{loss rate}\}$ is close to 0). In such scenarios, ideally one would like to find the sparsest solution to $\mathbf{y} = A\mathbf{x}$ by minimizing the L_0 norm $\|\mathbf{x}\|_0$ (i.e., the number of nonzeros in \mathbf{x}). But since the L_0 norm is not convex and is notoriously difficult to minimize, one often approximates L_0 norm with an L_1 norm. As shown in [19], the minimal L_1 norm solution often coincides with the sparsest solution for under-determined linear systems. We have successfully applied L_1 norm minimization to network anomaly inference [64].

As in [64], we solve the following variant of (7.21)

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 + \lambda \|\mathbf{x}\|_1. \quad (7.22)$$

An advantage of (7.22) is that it can be cast into an equivalent Linear Programming (LP) problem, for which solutions are available even with large-scale A , owing to modern interior-point LP methods. The LP formulation also allows one to incorporate additional linear constraints, such as the nonnegativity constraints $\mathbf{x} \geq 0$. Finally, if we have prior information that \mathbf{x} is close to an initial solution μ , we can replace $\|\mathbf{x}\|_1$ with $\|\mathbf{x} - \mu\|_1$ in (7.22), yielding

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 + \lambda \|\mathbf{x} - \mu\|_1. \quad (7.23)$$

7.2.3 Maximum Entropy Estimation

For inference under the nonnegativity constraints $\mathbf{x} \geq 0$, another commonly used solution is *maximum entropy estimation*, which uses the negative entropy function

as $J(\mathbf{x})$ in (7.18).

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \sum_i x_i \log x_i, \quad \mathbf{x} \geq 0. \quad (7.24)$$

If we know \mathbf{x} is close to an initial solution $\mu = [\mu_1 \mu_2 \cdots \mu_n]^T$, we can instead minimize the relative entropy [15], resulting in

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \sum_i x_i \log \frac{x_i}{\mu_i}, \quad \mathbf{x} \geq 0. \quad (7.25)$$

(7.25) can be efficiently solved by PDCO [51], which has been highly optimized for sparse matrices A . We have successfully applied (7.25) in the context of traffic matrix estimation [67].

7.3 Toolkit Development

We develop a toolkit on the PlanetLab [48] to measure and infer network path properties. Our toolkit is programmed in MATLAB, Perl, and C++, altogether with around 25,000 lines of code. The toolkit design is based on master-slave model. The master accepts measurement requests from users, designs measurement experiments, issues measurement commands to the slaves, and collects and analyzes the results. The slaves accept measurement commands from the master, conduct measurements, gather the results and return them back to the master. While our current implementation is based on one master and multiple slaves, our architecture is extensible to multiple masters and multiple slaves.

As shown in Figure 7.4, the master consists of the following five modules: (i) controller, which accepts user measurement requests, schedules jobs, and manages the other master modules, (ii) topology manager, which generates and maintains the routing matrix, (iii) experimenter, which applies one of the experimental design algorithms in Section 7.1 to identify which paths to probe, and issues measurement requests to the corresponding slave nodes, (iv) analyzer, which applies one of the inference algorithms in Section 7.2 to infer the network performance based on the obtained measurement data, and (v) communication module, which takes care of communication between slaves and masters.

Slave side of the toolkit accepts measurement commands both from topology

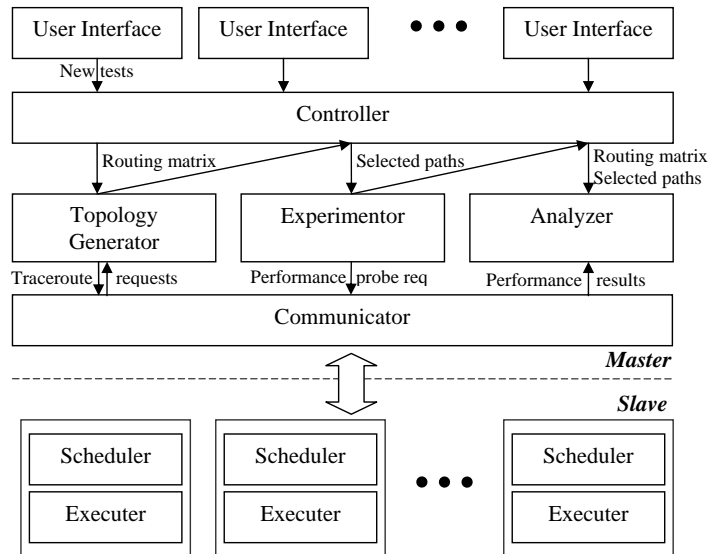


Figure 7.4: Our toolkit architecture.

manager and experimenter. The request is queued at its scheduler, and executed according to the specified frequency and duration. When a set of measurement experiments has finished, the results are sent back to the communication module of the master. For safety and convenience, we use scriptroute [56] for conducting traceroute. The toolkit runs continuously to measure and infer performance on the paths specified by the users.

Chapter 8

Evaluation

8.1 Evaluation Methodology

8.1.1 Accuracy Metric

We quantify the inference accuracy using normalized mean absolute error (MAE), which is defined as

$$\text{normalized } MAE = \frac{\sum_i |inferred_i - actual_i|}{\sum_i actual_i}, \quad (8.1)$$

where $inferred_i$ and $actual_i$ are the inferred and actual end-to-end performance for path i . A lower value of normalized MAE indicates better accuracy.

8.1.2 Dataset Description

We evaluate our approach using both real traces and synthetic data. *Note that the real traces use round-trip performance measurements, whereas the synthetic data use one-way performance measurements.* As noted in Section 3, the problem formulation $\mathbf{y} = \mathbf{A}\mathbf{x}$ works for both one-way and round-trip measurements.

PlanetLab traces: We collected RTT traces among PlanetLab using our toolkit on Oct. 1, 2005 for 10 minutes, with a probing frequency of one probe per second for every monitored path. We also collected loss traces on PlanetLab on Jan. 22, 2006 for 15 minutes, with a probing frequency of one probe per 300 milliseconds for every monitored path. Table 8.1 summarizes the traces, where *nodes* include both

	# nodes	# overlay nodes	# paths	# links	rank(A)
PlanetLab-RTT	2514	61	3657	5467	769
PlanetLab-loss	1795	60	3270	4628	690

Table 8.1: Summary of PlanetLab traces used for evaluation.

	# nodes	# overlay nodes	# paths	# links	rank(A)
Brite-n1000-o200	1000	200	39800	2883	2051
Brite-n5000-o600	5000	600	359400	14698	9729

Table 8.2: Summary of synthetic data used for evaluation.

end hosts and intermediate routers on the end-to-end paths, and *overlay nodes* only include end hosts among which the end-to-end performance needs to be monitored or estimated.

We construct routing matrix, A , using traceroute measurements. We derive the actual RTT based on the mean over 60 probes in every 1-minute interval, and derive the actual loss rates based on the mean over 300 probes in every 90-second interval. We use the following approach to derive the inferred RTT and loss rates: for the paths that are monitored we assume we know the true RTT and loss rates; for the un-monitored paths, we use the inference algorithms described in Section 7.2 to infer based on the observed performance of monitored paths. For each interval, we use normalized *MAE* to quantify the inference error.

Synthetic data: To further test the scalability of our approach, we generate synthetic network topologies using BRITE topology generator [37]. Table 8.2 summarizes the topologies we use. BRITE topology generator assigns each link with a delay based on its physical distance. In addition, we further assign a loss rate to each link in the following way as in [10, 45]. A fraction of links were classified as “good”, and the rest as “bad”. The loss rate for a good link is picked uniformly at random in the 0-1% range and that for a bad link is picked in the 5-10% range. Once each link has been assigned a loss rate, we derive the true loss rate for each path. Then we use Bernoulli or Gilbert loss process at each path to derive the observed loss rate. In the Bernoulli model, each packet is dropped with a fixed probability determined by the loss rate of the path. In the Gilbert case, the path moves between a good state and a bad state, where no packets are dropped at the good state and all packets are dropped at the bad state. Following [41, 45], we use 35% as the probability of remaining in the bad state. The other state-transition probabilities

are determined to match the average loss rate with the loss rate assigned to the link. In both cases, the end-to-end loss rate is computed based on the transmission of 10000 packets. Our evaluation shows that the inference accuracy is similar under Bernoulli and Gilbert loss models. So in the interest of brevity, we only show the results from Gilbert loss models.

8.1.3 Experimental Parameters

There are several parameters in Bayesian experimental design and network inference. Below we present the values that we use for these parameters in our evaluation.

Prior information for Bayesian experimental design (R): Recall that the design criteria for both Bayesian A - and D -optimality are functions of $D(\eta) = (A_S A_S^T + R)^{-1}$. To estimate R , one needs to estimate both the variance of the measurement noise (σ^2) and the covariance matrix of the link performance ($\Sigma = \sigma^2 R^{-1}$) through network measurement. However, the estimation of such second-order statistics in large-scale network measurement can be both expensive and inaccurate (due to measurement artifacts and non-stationarities in Internet path properties).

In NetQuest, we avoid such difficulties by setting $R = \epsilon I$, where ϵ is a small constant and I is the identity matrix. Our results suggest that this simple choice of R yields designs that consistently out-perform the alternative designs we considered (see Section 8.2). Moreover, the resulting design is highly insensitive to the choice of ϵ . In our evaluation, we set $\epsilon = 0.001$, which yields good results. Finally, note that a similar approach has been taken in the literature of Bayesian supersaturated design [34].

Prior information for network inference (μ): In this work, unless noted otherwise, we assume no prior information about \mathbf{x} . That is, we set $\mu = \mathbb{0}$ (an all-0 vector) for L_2 and L_1 norm minimization (both with and without nonnegativity constraints), and $\mu = \mathbb{1}$ (an all-1 vector) for maximum entropy estimation. Despite not using any prior information, our results show that NetQuest can achieve high accuracy by probing only a small fraction of paths.

In our future work, we plan to develop light-weight techniques to obtain better priors and thus further improving the accuracy. As an initial step, in Section 8.2.3 we evaluate a simple enhanced prior that does not require generating any extra probing traffic.

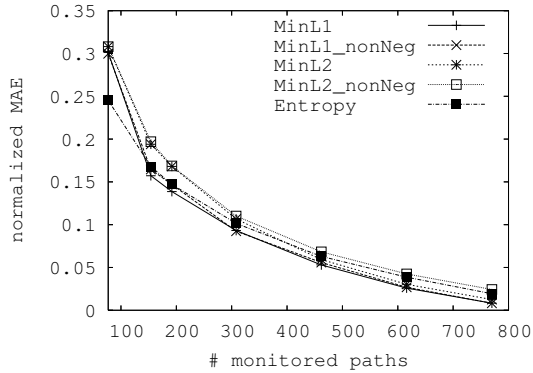


Figure 8.1: Comparison of inference algorithms for delay estimation in PlanetLab-RTT using A-optimal design.

Regularization parameter (λ): Our experience [67, 64] suggests that the inference algorithms are not sensitive to the choice of λ . In our evaluation, we use $\lambda = 0.01$, which gives satisfactory results.

8.2 Evaluation Results

We first evaluate our basic measurement framework. Then we examine its capability of supporting flexible design requirements. Finally we study the effects of prior information.

8.2.1 Basic Framework

In this section, we evaluate the two key components of our framework: (i) inference algorithms, and (ii) design of experiments.

Comparison of Inference Algorithms

First, we compare the accuracy of different inference algorithms. We use the A-optimal design criterion to determine which set of paths to monitor. Figure 8.1 and Figure 8.2 show the error of inferring end-to-end delay and loss rates as we vary the number of monitored paths. The x-value of the right most point on each curve corresponds to the rank of the routing matrix.

As shown in Figure 8.1, different inference algorithms perform similarly for

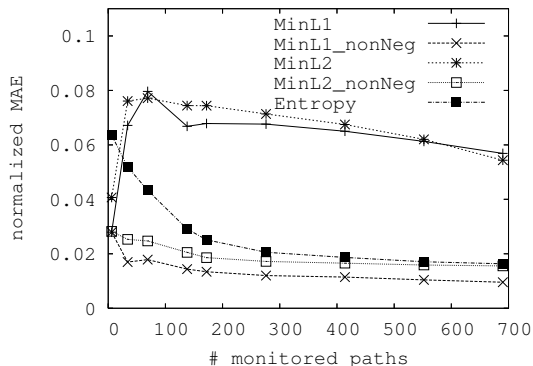


Figure 8.2: Comparison of inference algorithms for loss estimation in PlanetLab-loss using A-optimal design.

delay inference. Moreover, as expected, the error decreases with an increasing number of monitored paths.

As shown in Figure 8.2, the performance difference between various inference algorithms is larger for loss rate inference. The inference algorithms that enforce nonnegativity constraints out-perform those that do not enforce such constraints. In addition, the inference error under those algorithms without nonnegativity constraints does not decrease with an increasing number of monitored paths. Since loss rates take nonnegative values, intuitively enforcing nonnegativity constraints should give better inference. In comparison, the effect of nonnegativity constraints is much smaller for delay inference. This is because all paths have delay larger than 0, so even without enforcing nonnegativity constraints most links are assigned positive delay. Finally, MinL1 consistently out-performs the other inference schemes. As described in Section 7.2.2, MinL1 effectively maximizes the sparsity of link loss rate, which matches well with the fact that few links on the Internet are lossy.

From the above results, in the rest of the thesis, unless noted otherwise, we use MinL1_nonNeg for both delay and loss inference.

Comparison of Measurement Designs

Next we evaluate different algorithms for designing measurement experiments. We consider inferring two types of quantities: network-wide performance and individual path performance.

Network-wide performance: A global view of the performance aggregated over

an entire network is useful for a variety of reasons. It can be used for estimating a typical user experience (as in the Internet End-to-end Performance Monitoring Project (IEPM)), detecting anomalies, trouble-shooting, and optimizing performance. The pioneering work in this area is done by Chua et al. [11, 12], which is based on SVD.

We compare the Bayesian experimental designs with the other alternatives for inferring network wide average delay. Here the quantity of interest is $f(\mathbf{x}) = \frac{1}{m} \mathbf{1} A \mathbf{x}$, where $\mathbf{1}$ is an all-1 row vector of length m . We use the PlanetLab traces to evaluate the performance under a realistic scenario, and use simulated topologies to further test the scalability of our measurement design. Figure 8.3 compares different experimental design schemes when MinL2 is used for inference, and Figure 8.4 shows the results when MinL1_nonNeg is used. As noted in Section 7.1.1, A -optimal and D -optimal designs are identical for inferring network-wide average, so we only show the results of A -optimal. A -optimal is more scalable than SVD and QR, both of which cannot handle Brite-n5000-o600, and fail to run on Brite-n1000-o200 when the number of monitored paths exceeds 1200. Therefore the SVD and QR curves in Figure 8.3(b) and Figure 8.4(b) stop at 1200 monitored paths, and Figure 8.3(c) and Figure 8.4(c) only show the results for Bayesian experimental designs and random path selection for Brite-n5000-o600.

As shown Figure 8.3(a)-(c), with the A -optimal design, the inference error decays very fast – the error is within 15% by monitoring only within 2% paths (e.g., 77 out of 3657 paths in PlanetLab-RTT, 409 out of 39800 paths in Brite-n1000-o200, 1945 out of 359400 paths in Brite-n5000-o600). In comparison, the inference error is 50% or higher (than A -optimal) when the same number of paths are monitored under the other schemes. In addition, we observe that to achieve within 10% inference error, the other schemes require monitoring 60% more paths than A -optimal. Finally, as the number of monitored paths increases, all the schemes converge to close to 0 inference error, since in this case there are sufficient information to reconstruct the global network view. Random selection converges slower because it does not ensure the selected paths are linearly independent. Similar results are observed in Figure 8.4 when the inference algorithm changes to MinL1_nonNeg.

Individual path performance: Figure 8.5 compares different measurement design schemes for inferring individual path delay. Note that the results for SVD and QR are not available for Brite-n1000-o200 over 1200 monitored paths, nor for

Brite-n5000-o600, since they do not scale well. As we can see, the rank-based approach requires monitoring 769 - 9729 paths, which is expensive. In comparison, the other approaches can provide a smooth tradeoff between inference accuracy and measurement overhead. Among them, *A*-optimal performs the best. To achieve 10% inference error, the other algorithms need to monitor up to 60% more paths than *A*-optimal. Note that *D*-optimal performs significantly worse than *A*-optimal, and sometimes even worse than the other alternatives. This is likely due to the fact that the Kullback-Leibler distance tends to under-penalize estimation errors. Finally, we observe that the performance benefit of *A*-optimal is largest when the number of monitored path is close to one to two thirds of the rank of the routing matrix. This is because when the number of monitored paths is too small, there is not enough information for accurately reconstructing the global view of network performance, regardless of which design scheme is used. In the other extreme, when the number of monitored paths is close to the rank, any independent row combinations (in the routing matrix) can provide sufficient information for accurate reconstruction of end-to-end path properties.

Figure 8.6 shows the absolute inference error in loss rate estimation. *A*-optimal slightly out-performs the other schemes. The performance gap is smaller than that of delay, because most links have close to zero loss rate, and assigning links with zero loss rate (even without extensive network monitoring) can achieve low average inference error. Note that when the number of monitored paths is equal to the rank of the routing matrix, the error is non-zero due to the sampling errors we introduce when assigning true loss rates in synthetic topologies. Finally, we observe that *D*-optimal is not competitive, and performs considerably worse than *A*-optimal. We observe similar results for Bernoulli loss model and other topologies. So hereafter we will focus on Bayesian *A*-optimal designs.

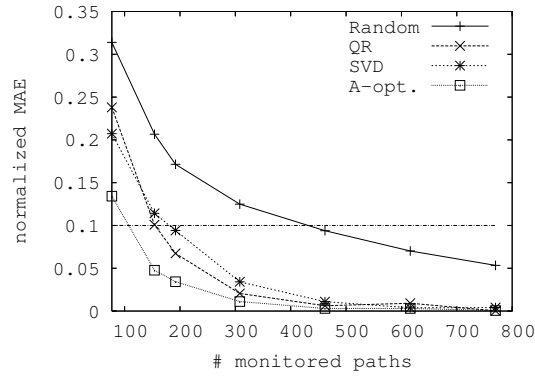
Summary

To summarize, in this section we evaluate our measurement framework. Our key findings are:

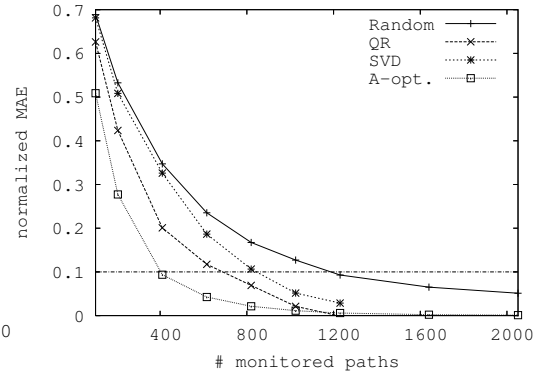
- Measurement design is crucial for large-scale network monitoring. *A*-optimal is effective in constructing measurement experiments for inferring network-wide average delay. It can achieve 15% inference error by monitoring only 2% paths.

Moreover it is also competitive for estimating individual path performance. In addition, it is highly scalable, and can be applied to networks with thousands of routers and end hosts.

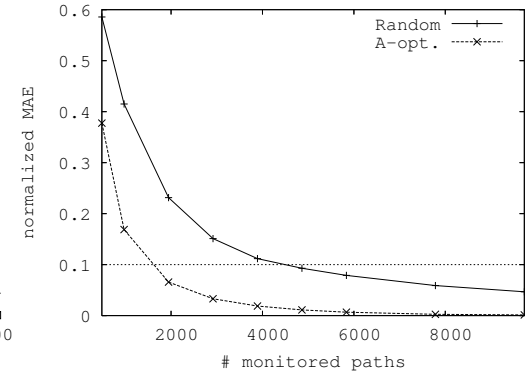
- Our results show that the different inference algorithms under study perform similarly for delay inference, whereas the algorithms that enforce nonnegativity constraints perform better for loss inference.



(a) PlanetLab-RTT

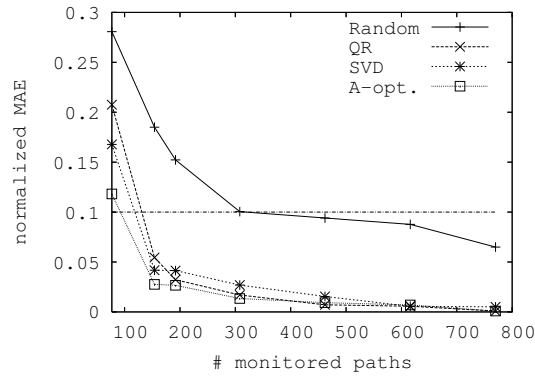


(b) Brite-n1000-o200

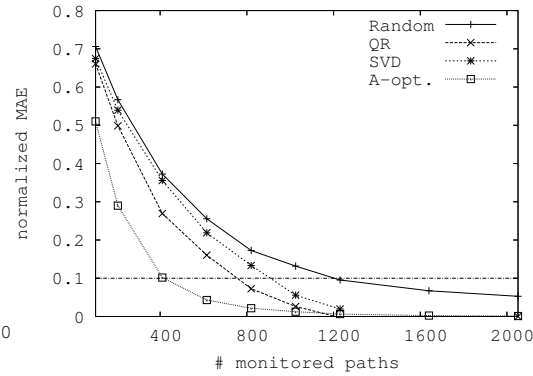


(c) Brite-n5000-o600

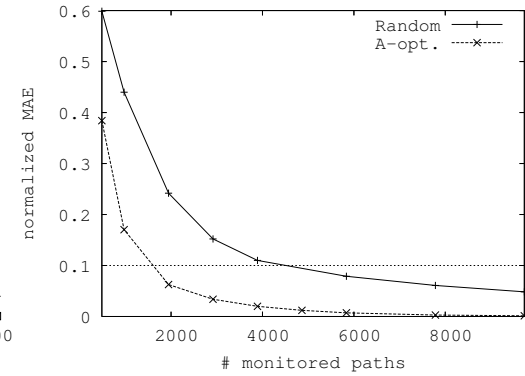
Figure 8.3: Comparison of experimental designs for estimating network-wide delay using MinL2 inference algorithm.



(a) PlanetLab-RTT

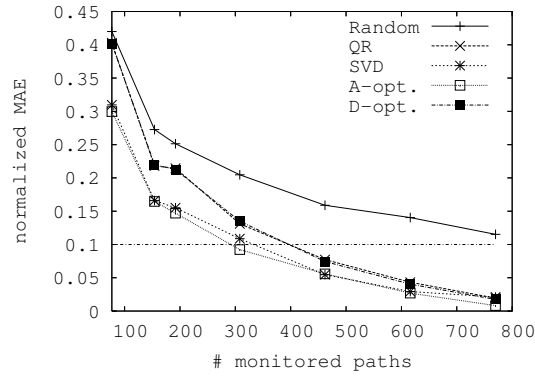


(b) Brite-n1000-o200

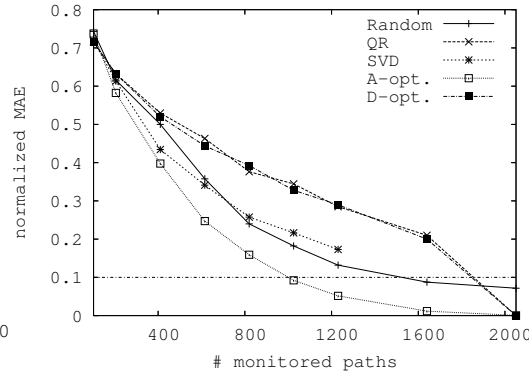


(c) Brite-n5000-o600

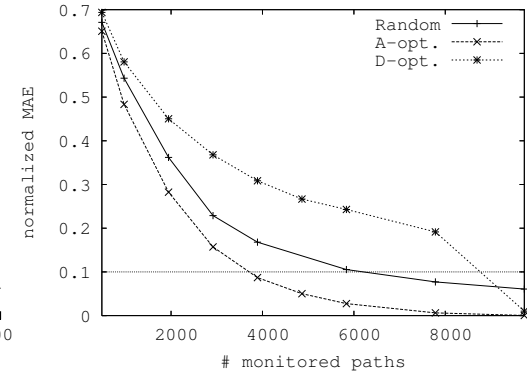
Figure 8.4: Comparison of experimental designs for estimating network-wide delay using MinL1_nonNeg inference algorithm.



(a) PlanetLab-RTT

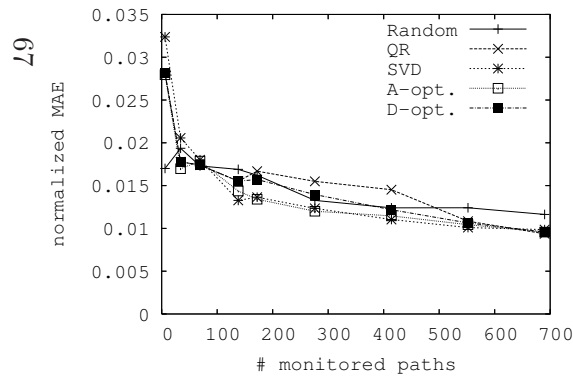


(b) Brite-n1000-o200

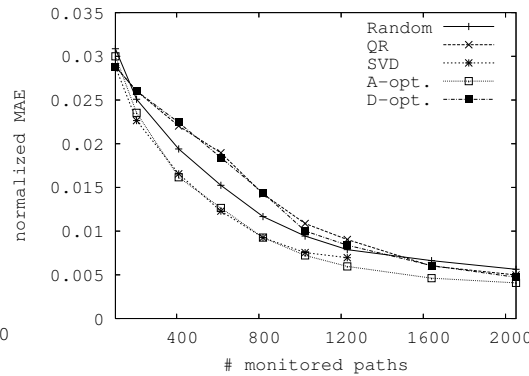


(c) Brite-n5000-o600

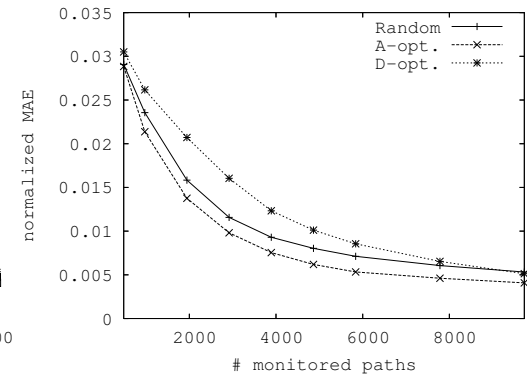
Figure 8.5: Comparison of experimental designs for per-path delay inference using MinL1_nonNeg inference algorithm.



(a) PlanetLab-loss



(b) Brite-n1000-o200



(c) Brite-n5000-o600

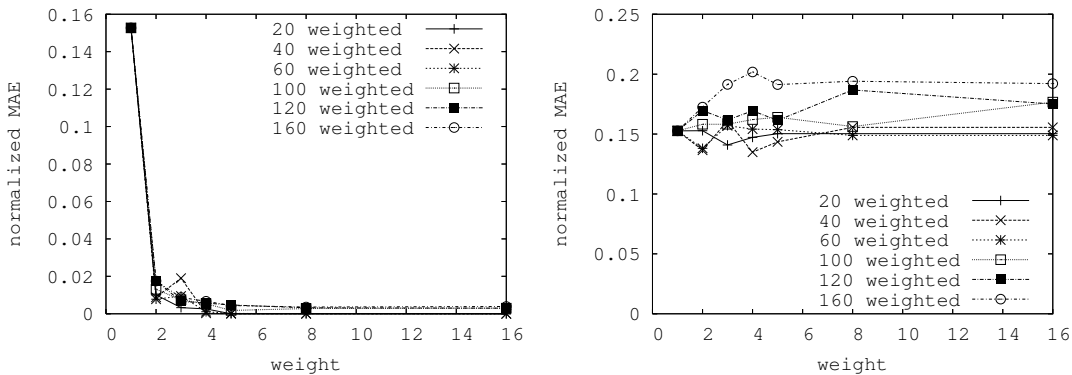
Figure 8.6: Comparison of experimental designs for per-path loss inference using MinL1_nonNeg inference algorithm (simulation uses the Gilbert loss model).

8.2.2 Flexibility of Measurement Design

In this section, we evaluate the flexibility of our measurement framework by estimating delay on individual network paths in the PlanetLab-RTT topology.

Differentiated Design

First we examine the effectiveness of experimental designs for providing differentiated treatment to a subset of paths. We apply the technique described in Section 7.1.2 to achieve this goal. In our evaluation, we randomly assign a subset of preferred paths with a weight varying from 1 to 16, while fixing the weight on the remaining paths to 1. Figure 8.7 shows the inference error on both the preferred and the remaining paths when we monitor 200 paths in PlanetLab-RTT topology and vary the number of preferred paths from 20 to 160. We make the following observations. First, as we would expect, the inference error on the preferred paths decreases with an increasing weight. When the weight is 4 and higher, the inference error is close to 0. This is because when the weight is high enough, the performance of many preferred paths is either directly monitored or exactly reconstructed from the monitored paths. Second, we observe that the inference error on the remaining paths increases slightly, since as we pay more attention to the preferred paths, the



(a) Inference error on the preferred paths (b) Inference error on the remaining paths

Figure 8.7: Use differentiated design based on A -optimal design to provide a higher resolution in estimating a selected set of preferred paths in PlanetLab-RTT.

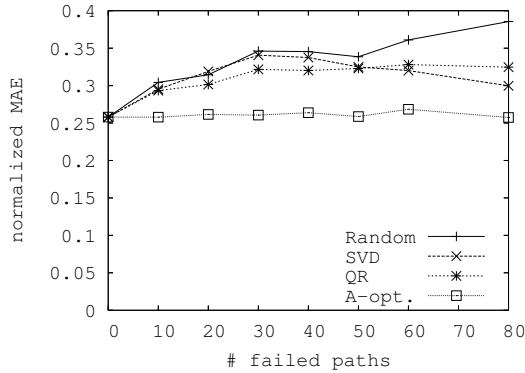


Figure 8.8: Comparison of various augmented design schemes in PlanetLab-RTT.

remaining paths are monitored less extensively. For a similar reason, the inference error of the remaining paths tends to increase slightly with an increasing number of preferred paths.

Augmented Design

Next we consider augmented design for supporting continuous monitoring. Our evaluation is based on the following scenario. Suppose we identify a set of paths to monitor, and some of them fail to provide us measurement data (e.g., due to software or hardware failures at monitor sites or at their incoming/outgoing links). In this case, we need to identify the additional measurements to conduct given that we have already obtained the measurement results from the unfailed paths.

In our evaluation, we first use *A*-optimal to identify 100 paths to monitor in PlanetLab-RTT. Then we vary the number of failed paths from 10 to 80, and apply different augmented design algorithms to determine the additional paths to monitor. Figure 8.8 shows the average inference error under different schemes. As we can see, *A*-optimal yields the lowest error. Moreover, its inference error is similar for a varying number of failed paths. In comparison, the inference error of the other schemes tends to increase with an increasing number of failed paths. This suggests that the *A*-optimal design is the most effective in augmenting existing designs.

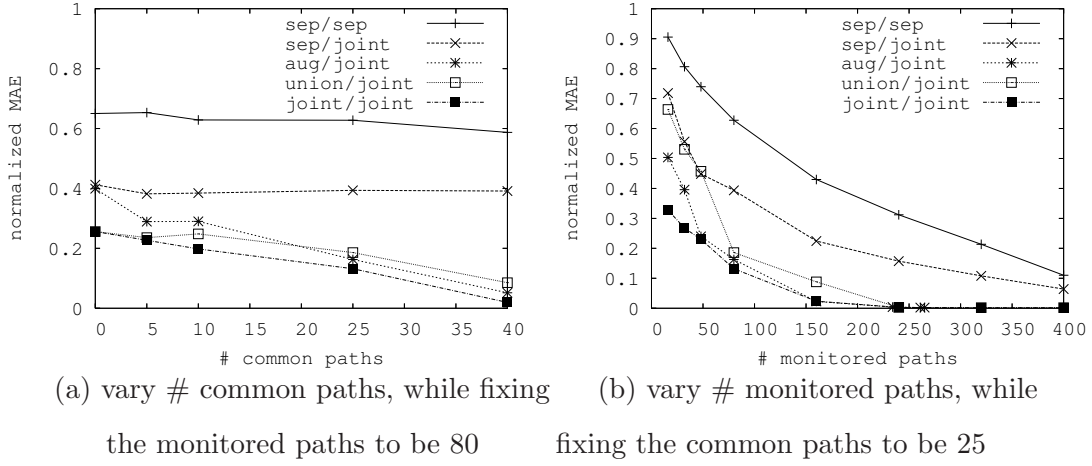


Figure 8.9: Comparison of different design modes in handling multi-user scenarios using PlanetLab-RTT, where all modes use the A -optimal design.

Multi-user Design

Now we study the multi-user scenarios, where each user is interested in a certain part of network. We compare the following design schemes:

- *Separate design and separate inference (sep./sep.):* Each user individually determines the set of paths to monitor, and makes inference based solely on his/her own observations.
- *Separate design and joint inference (sep./joint):* This is an enhancement of the previous version. Users still individually decide which paths to monitor, but they make inference based on the observations made from all users.
- *Augmented design and joint inference (aug./joint):* In the augmented design, we first design measurement experiments for user 1, and then apply the augmented design (in Section 8.2.2) to construct measurement experiments for user 2. We continue the process for all the other users.
- *Union design and joint inference (union/joint):* In the union design, we take a union of all the paths that are interesting to at least one user, and then apply the (basic) measurement design.

- *Joint design and joint inference (joint/joint)*: Unlike in the union design, where all interesting paths are treated equally, in joint design we set a path’s weight to be the square root of the number of users who are interested in the path (see Section 7.1.2).

All the design algorithms can work in separate, augmented, and union modes. In addition, *A*-optimal can also support joint design.

In our evaluation, we have 16 users, each interested in 50 paths. There are a common set of paths that are interesting to all users. Figure 8.9(a) compares the *A*-optimal design in its various design modes for inferring individual path delay as the number of common paths is varied from 0 to 40. The remaining paths interesting to a user are randomly selected from all the non-common paths. In the order of accuracy ranking, joint/joint > union/joint \approx aug./joint > sep./joint > sep./sep. In particular, sep./sep. incurs significantly higher error than the others, because in this case different users do not share their observation. Enabling information sharing in sep./joint reduces the normalized *MAE* by 0.2 or higher. A further reduction is achieved by incorporating users’ interest into measurement design. Interestingly, augmented design performs similarly to union design, even though the former is an online version of the latter (i.e., the *i*-th user determines its measurement experiments without considering the *j*-th user’s interest for $j > i$). The performance of joint/joint is even better, and its benefit over separate design grows as the number of common paths increases. This is attributed to the fact that it not only incorporates all users’ interest in designing measurement, but also it biases measurement towards paths that interest more users. Figure 8.9(b) compares the inference error as we vary the number of monitored paths. As before, joint/joint yields the best performance among all the schemes. The performance gap is largest with a small number of monitored paths. This suggests that experimental design is especially important under tight measurement resource constraints.

Next we compare the performance across different design schemes. Figure 8.10 (a) and (b) show the inference error of various design schemes under their best design modes. They all use joint inference. As they show, *A*-optimal yields the lowest error, up to 80% lower than the alternatives.

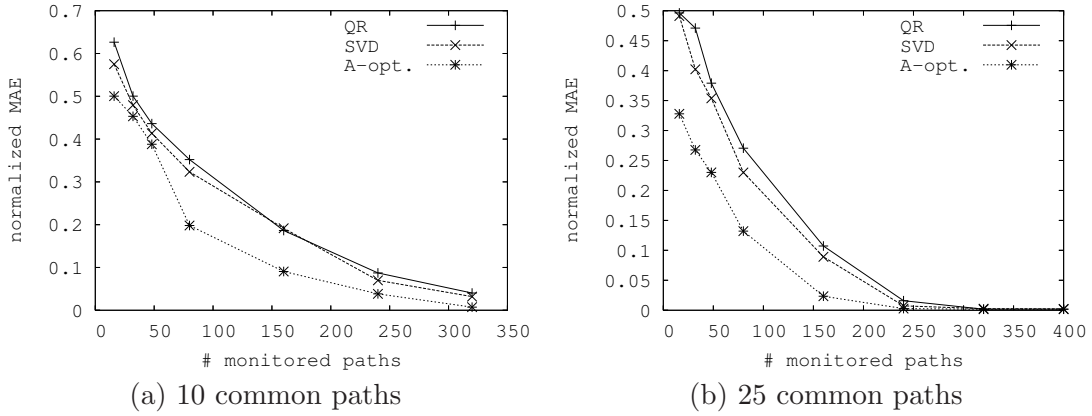


Figure 8.10: Comparison of different design schemes using their best modes on PlanetLab-RTT.

Summary

To summarize, we demonstrate the flexibility of our measurement framework using the real trace. Our results show that it can effectively support differentiated design, augmented design, and joint design. Such capabilities are useful for a variety of network monitoring applications.

8.2.3 Effects of Prior Information

So far we assume no prior information about \mathbf{x} . In our future work, we plan to develop light-weight techniques to obtain better priors and thus further improving the inference accuracy. As a first step, below we develop a simple method for obtaining an enhanced prior. Specifically, we consider a special form of prior whose elements are all equal: $\mu = z \cdot \mathbf{1}$, where z is an unknown, and $\mathbf{1}$ is an all-1 column vector of length n . We can estimate z by solving an over-determined least-squares problem: $\mathbf{y} = A\mu = (A\mathbf{1})z$, yielding $z = \frac{(A\mathbf{1})^T \mathbf{y}}{\|A\mathbf{1}\|_2^2}$. Intuitively, the resulting prior $\mu = \frac{(A\mathbf{1})^T \mathbf{y}}{\|A\mathbf{1}\|_2^2} \cdot \mathbf{1}$ estimates the average link performance. An advantage of this method is that it is extremely simple and requires no extra measurement.

Figure 8.11 shows the accuracy of delay inference for different measurement design schemes using the above enhanced prior. Compared with Figure 8.5(a), we make the following observations. First, the enhanced prior improves the inference

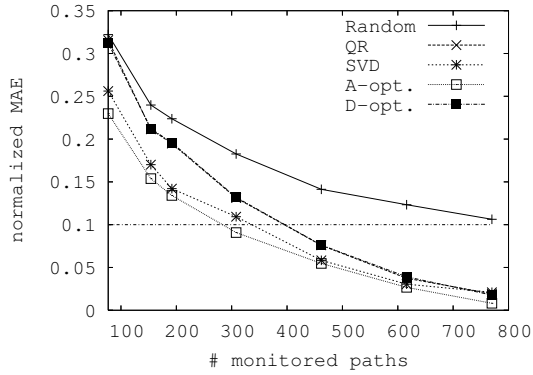


Figure 8.11: Effects of the enhanced prior on PlanetLab-RTT.

accuracy for all measurement design schemes. It reduces the normalized *MAE* by up to 0.07 (or about 25%). The improvement is largest when the number of monitored paths is small. This is because the enhanced prior information is most helpful to compensate for incomplete monitoring information. In comparison, with extensive monitoring we can accurately estimate performance even without prior. Second, the relative ranking of different design schemes remains the same as before. A-optimal design continues to yield the highest accuracy.

On the other hand, the enhanced prior only yields very little accuracy improvement for loss inference (results omitted in the interest of brevity). This is not surprising, because most links have very low loss rate, making $\mu = \mathbb{0}$ a good prior.

Chapter 9

Summary

In this work, we develop NetQuest, a flexible framework for large-scale network measurement and inference. It consists of two major components: the design of measurement experiments, and network inference. For the former, we leverage powerful tools developed in the field of Bayesian experimental design. For the latter, we build on top of a number of existing network tomography techniques to infer network properties based on partial, indirect observations. Our framework is flexible, and can accommodate a variety of design objectives, such as differentiated, augmented, and multi-user designs. It is also highly scalable and can design measurement experiments that span thousands of routers and end hosts.

Chapter 10

Conclusion and Future Work

In this thesis, we endeavored to build a scalable and flexible network measurement framework. We conclude this thesis by summarizing major contributions and discussing avenues for future research.

10.1 Conclusion

We make the following contributions.

- For an overlay network with n end hosts, we show that the upper-bound of number of paths to fully describe all the $O(N^2)$ grows as $O(N \log N)$.
- By leveraging Bayesian experimental design tools in the field of large-scale network measurement, we earn a very good accuracy with a small monitoring cost. The efficiency of the monitoring enables a good scalability for large-scale network measurement.
- We build the unified framework on top of Bayesian experimental design and inference techniques to infer network properties based on partial, indirect observations. The framework is flexible to accommodate differentiated, augmented, and multi-user designs.
- We develop a toolkit that implements our framework on a real Internet environment. With the toolkit we conduct an extensive evaluation of our framework for efficient monitoring of end-to-end network performance.

10.2 Future Work

There are several avenues for future work.

First, we plan to further enhance the robustness of our experimental design and inference by making the design *fault tolerant*. Specifically, we want a design that minimizes the *worst-case* design criterion in the presence of multiple faulty paths (*i.e.*, paths that experience either failures or major routing changes).

Second, we are interested in applying our techniques to estimating other network properties, such as traffic matrix estimation. In particular, we may use Bayesian experimental design to identify strategic locations to place additional measurement capabilities to enhance the accuracy of traffic matrix estimation.

Third, we would like to extend our framework to incorporate additional design constraints and to handle non-linear metrics. Note that there is a rich literature on non-linear Bayesian experimental design [6].

Finally, we are interested in developing light-weight techniques to obtain better prior information.

Bibliography

- [1] E. Anderson et al. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.
- [2] E. AWF. *Likelihood*. Cambridge University Press, 1972.
- [3] R. Barrett et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.
- [4] T. Bu, N. Duffield, F. Presti, and D. Towsley. Network tomography on general topologies. In *ACM SIGMETRICS*, 2002.
- [5] R. Caceres, N. G. Duffield, J. Horowitz, D. Towsley, and T. Bu. Multicast-based inference of network internal characteristics: Accuracy of packet loss estimation. In *Proc. of IEEE INFOCOM*, 1999.
- [6] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:273–304, 1995. <http://citeseer.ist.psu.edu/chaloner95bayesian.html>.
- [7] Y. Chen. *Towards a Scalable, Adaptive and Network-aware Content Distribution Network*. PhD thesis, University of California at Berkeley, Nov. 2003.
- [8] Y. Chen, D. Bindel, and R. H. Katz. Tomography-based overlay network monitoring. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, 2003.
- [9] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 55–66, New York, NY, USA, 2004. ACM Press.

- [10] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *Proc. of ACM SIGCOMM*, 2004.
- [11] D. B. Chua, E. D. Kolaczyk, and M. Crovella. Efficient monitoring of end-to-end network properties. In *Proc. of IEEE INFOCOM*, Jul. 2004.
- [12] D. B. Chua, E. D. Kolaczyk, and M. Crovella. A statistical framework for efficient monitoring of end-to-end network properties. In *Proc. of ACM SIGMETRICS*, 2005.
- [13] M. A. Clyde. Experimental design: A Bayesian perspective. *International Encyclopedia Social and Behavioral Sciences*, Apr. 2001.
- [14] M. Coates, A. Hero, R. Nowak, and B. Yu. Internet Tomography. *IEEE Signal Processing Magazine*, 19(3):47–65, 2002.
- [15] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: Wiley, 1991.
- [16] T. Davis. LDL: A sparse LDL' factorization and solve package (version 1.2), 2005. <http://www.cise.ufl.edu/research/sparse/ldl/>.
- [17] J. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.
- [18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood for incomplete data via EM algorithm (with discussion). *Journal of the Royal Statistical Society (Series B)*, 39, 1977.
- [19] D. Donoho. For most large underdetermined systems of linear equations, the minimal l_1 -norm near-solution approximates the sparsest near-solution. <http://www-stat.stanford.edu/~donoho/Reports/2004/l1l0approx.pdf>.
- [20] D. Donoho. For most large underdetermined systems of linear equations, the minimal l_1 -norm solution is also the sparsest solution. <http://www-stat.stanford.edu/~donoho/Reports/2004/l1l0EquivCorrected.pdf>.
- [21] D. Donoho and M. Elad. Maximal sparsity representation via l_1 minimization. *Proc. Nat. Aca. Sci.*, 100:2197–2202, March 2003.

- [22] N. Duffield, J. Horowitz, and F. L. Presti. Adaptive multicast topology inference. In *Proc. IEEE Infocom 2001*, Apr. 2001.
- [23] B. Efron and C. Morris. Stein's paradox in statistics. *Scientific American*, 236(5):119–127, 1977.
- [24] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationship of the Internet topology. In *ACM SIGCOMM*, 1999.
- [25] V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.
- [26] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *In IEEE Trans. Pattn. Anal.*, Mar. 1984.
- [27] W. R. Gilks, S. Richardson, and D. J. Spiegelalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.
- [28] G. Golub and C. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989.
- [29] G. H. Golub, V. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Technical Report CS-TR-76-559, Stanford University, Aug. 1976. <http://www-db.stanford.edu/TR/CS-TR-76-559.html>.
- [30] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 2nd edition, 1989.
- [31] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *IEEE INFOCOM*, 2000.
- [32] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, 1997.
- [33] A. Hoerl and R. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(3):55–67, 1970.
- [34] B. Jones, D. Lin, and C. Nachtsheim. Bayesian D-optimal supersaturated designs, 2004. Submitted for publication.
- [35] E. Lehmann. *Theory of Point Estimates*. Wiley, New York, 1983.

- [36] D. V. Lindley. *Bayesian Statistics – A Review*. SIAM, Philadelphia, PA, 1972.
- [37] A. Medina, A. Lakhina, I. Matta, and J. Byers. Boston university representative Internet topology generator (BRITE). <http://www.cs.bu.edu/brite/>.
- [38] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proc. of ACM SIGCOMM*, Aug. 2002.
- [39] A. J. Miller and N. K. Nguyen. A Fedorov exchange algorithm for D-optimal design. *Applied Statistics*, 1994.
- [40] T. J. Mitchell. An algorithm for the construction of D-optimal designs. *Technometrics*, 1974.
- [41] E. M. Nahum, C. C. Rosu, S. Seshan, and J. Almeida. The effects of wide-area conditions on WWW server performance. In *Proc. of SIGMETRICS*, Jun. 2001.
- [42] N. K. Nguyen and A. J. Miller. A review of exchange algorithms for constructing discrete D-optimal designs. *Computational Statistics and Data Analysis*, 1992.
- [43] H. C. Ozmutlu et al. Managing end-to-end network performance via optimized monitoring strategies. *Journal of Network and System Management*, 10(1), 2002.
- [44] V. Padmanabhan, L. Qiu, and H. Wang. Server-based inference of Internet link lossiness. In *IEEE INFOCOM*, 2003.
- [45] V. N. Padmanabhan, L. Qiu, and H. Wang. Server-based inference of Internet performance. In *Proc. of IEEE INFOCOM*, Mar. 2003.
- [46] V. Paxson. End-to-end Internet packet dynamics. In *ACM SIGCOMM*, 1997.
- [47] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5), 1997.
- [48] Planetlab. <http://www.planet-lab.org>.
- [49] RON measurement data. <http://nms.lcs.mit.edu/ron/data/>.

- [50] D. Rubenstein, J. F. Kurose, and D. F. Towsley. Detecting shared congestion of flows via end-to-end measurement. *ACM Transactions on Networking*, 10(3), 2002.
- [51] M. Saunders. PDCO: Primal-Dual interior method for Convex Objectives, 2003. <http://www.stanford.edu/group/SOL/software/pdco.html>.
- [52] Y. Shavitt, X. Sun, A. Wool, and B. Yener. Computing the unmeasured: An algebraic approach to Internet mapping. In *IEEE INFOCOM*, 2001.
- [53] H. H. Song, L. Qiu, and Y. Zhang. Netquest: A flexible framework for large-scale network measurement. In *Proc. of SIGMETRICS*, Jun. 2006.
- [54] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rockefuel. In *ACM SIGCOMM*, 2002.
- [55] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A facility for distributed internet measurement. In *USITS*, 2003.
- [56] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A facility for distributed internet measurement. In *Proc. of USITS*, Mar. 2003.
- [57] G. W. Stewart. *Matrix Algorithms: Basic Decompositions*. Society for Industrial and Applied Mathematics, 1998.
- [58] L. Subrmanian, S. Agarwal, J. Rexford, and R. H.Katz. Characterizing the Internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.
- [59] C. Tang and P. McKinley. On the cost-quality tradeoff in topology-aware overlay path probing. In *IEEE ICNP*, 2003.
- [60] H. Tangmunarunkit et al. Network topology generators: Degree-based vs structural. In *ACM SIGCOMM*, 2002.
- [61] R. Tibshirani. The Lasso page. <http://www-stat.stanford.edu/~tibs/lasso.html>.
- [62] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

- [63] Y. Zhang et al. On the constancy of Internet path properties. In *Proc. of SIGCOMM IMW*, 2001.
- [64] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proc. Internet Measurement Conference*, Oct. 2005.
- [65] Y. Zhang, V. Paxson, and S. Shenker. The stationarity of Internet path properties: Routing, loss, and throughput. ACIRI Technical Report, May, 2000.
- [66] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *Proc. of ACM SIGMETRICS*, Jun. 2003.
- [67] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. of ACM SIGCOMM*, Aug. 2003.

Vita

Han Hee Song was born in Tokyo, Japan on October 5, 1978, to Dr. Bang Ho Song and Mrs. Sook Ja Lee. He graduated from Kyungpook National University's Attached High School, Daegu, Korea in 1997. He received the degree of Bachelor of Science in Computer Sciences from Yonsei University in August, 2004. Prior to graduation, he attended University of California, Berkeley for a year in 2003.

From 1999 to 2001, he was employed at Dong-Suh Industrial Development Co., Ltd. in Seoul, Korea as software engineer. In 2002 and 2003, he was employed at Soft-On-Net, Inc. in Seoul, Korea and Soft-On-Net Japan, Inc. in Tokyo, Japan as software developer.

Permanent Address: Sampoong Apt. 5-532, Doosan-Dong
Soosong-Gu, Daegu, 706-752
Republic of Korea
hhsong@cs.utexas.edu

This thesis was typeset with L^AT_EX 2_ε¹

¹L^AT_EX 2_ε is an extension of L^AT_EX. L^AT_EX is a collection of macros for T_EX. T_EX is a trademark of the American Mathematical Society.