UNIVERSITY OF CALIFORNIA, SAN DIEGO

Efficient Gain-Driven Routing-Assisted Mechanisms for Network-wide Traffic Monitoring

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Communication Theory and Systems)

by

Chia-Wei Chang

Committee in charge:

Professor Bill Lin, Chair
Professor Sujit Dey
Professor Massimo Franceschetti
Professor Tajana Simunic Rosing
Professor Geoffrey M. Voelker

2011

The dissertation of Chia-Wei Chang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____
Chair

University of California, San Diego

2011

DEDICATION

To my dear father, Chung-Shin Chang, and mother,Yu-Ing Chang, who have waited patiently for this day to arrive.

EPIGRAPH

Stay Hungry. Stay Foolish. And I have always wished that for myself.
And now, as you graduate to begin anew, I wish that for you.
—Steve Jobs

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to deeply thank my advisor, Prof. Bill Lin, for his encouragement, brilliant guidance and support. His breadth of knowledge and intuition will always inspire me. Next, I would like to thank my co-authors, Guanyao Huang, Han Liu and Prof. Chen-Nee Chuah for their help and contributions towards this dissertation. I am grateful to my dissertation committee members, Prof. Sujit Dey, Prof. Massimo Franceschetti, Prof. Tajana Simunic Rosing and Prof. Geoffrey M. Voelker for their useful comments and words of advice. I would also like to thank my mentors, Dr. Jia Wang, Dr. Seungjoon Lee, Dr. Subhabrata Sen, Dr. Oliver Spatscheck and Mr. Alexandre Gerber for their guidance and instructions during my internships in AT&T Labs Research. Following, I want to thank my colleagues and friends at UCSD, specially Jerry Chou, Rohit Sunkam Ramanujam, Hao Wang, Shan Yan, Somsak Kittipiyakul, and Jaewook Shim for making this journey through graduate school fun and memorable. Finally, I would like to thank my parents and everyone in my family who stood behind me for all these years and I consider myself extremely lucky to have their love and support.

Chapter 2, in full, is a reprint of the material as it appears in IEEE Transactions on Network and Service Management (TNSM) 2011, Guanyao Huang, Chia-Wei Chang, Chen-Nee Chuah and Bill Lin, “Measurement-aware Monitor Placement and Routing: A Joint Optimization Approach for Network-Wide Measurements”. The dissertation author was the primary investigator and second author of the paper.

Chapter 3, in part, is a reprint of the material as it appears in the following publications:

- Chia-Wei Chang, Guanyao Huang, Bill Lin and Chen-Nee Chuah, “LEISURE: A Framework for Load-Balanced Network-Wide Traffic Measurements”, *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*,Brooklyn, NY, October 3-4, 2011.

Chapter 3, in full, has been submitted for publication of material as it may appear in IEEE Transactions on Network and Service Management (TNSM), Chia-Wei Chang, Guanyao Huang, Bill Lin and Chen-Nee Chuah, “A Joint Optimization Approach for

Load-Balanced Network-Wide Traffic Measurements and Monitor Placement". The dissertation author was the primary investigator and author of the papers.

Chapter 4, in part, is a reprint of the material as it appears in in the following publications:

- Chia-Wei Chang, Han Liu, Guanyao Huang, Bill Lin, and Chen-Nee Chuah, "Distributed Measurement-Aware Routing: Striking a Balance between Measurement and Traffic Engineering", *IEEE 31st International Conference on Computer Communications (INFOCOM)*, Orlando, Florida, March 25-30, 2012.

Chapter 4, in full, has been submitted for publication of material as it may appear in IEEE Transactions on Networking (ToM), Chia-Wei Chang, Han Liu, Guanyao Huang, Bill Lin and Chen-Nee Chuah, " Distributed Measurement-Aware Routing for Multiple Classes of Flows". The dissertation author was the primary investigator and author of the papers.

VITA

2002	Bachelor of Science in Communications Engineering, National Chiao Tung University
2004	Master of Science in Communications Engineering, National Chiao Tung University
2006-2011	Graduate Research Assistant, University of California, San Diego
2011	Doctor of Philosophy in Electrical Engineering (Communication Theory and Systems), University of California, San Diego

PUBLICATIONS

Chia-Wei Chang and Bill Lin, "A Simple Mechanism for Throttling High-Bandwidth Flows", *Research Letters in Communications (RLC)*, Vol. 2008, Article ID 704878, Oct 2008.

Chia-Wei Chang, Seungjoon Lee, Bill Lin and Jia Wang, "The Taming of The Shrew: Mitigating Low-Rate TCP-Targeted Attack", *IEEE 29th International Conference on Distributed Computing Systems(ICDCS)*, Montreal, Canada, June 2009. (acceptance rate=16%)

Chia-Wei Chang, Bill Lin, Alexandre Gerber, Subhabrata Sen and Oliver Spatscheck, "Network DVR: A Programmable Framework for Application-Aware Trace Collection", *Passive and Active Measurement Conference (PAM)*, Zurich, Switzerland, April 7-9, 2010. (acceptance rate=29%)

Chia-Wei Chang, Guanyao Huang, Bill Lin and Chen-Nee Chuah, "LEISURE: A Framework for Load-Balanced Network-Wide Traffic Measurements", *ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)*, Brooklyn, NY, October 3-4, 2011.

Chia-Wei Chang, Han Liu, Guanyao Huang, Bill Lin and Chen-Nee Chuah, " Distributed Measurement-Aware Routing: Striking a Balance between Measurement and Traffic Engineering", *IEEE 31st International Conference on Computer Communications (INFOCOM)*, Orlando, Florida, March 25-30, 2012.

Chia-Wei Chang, Seungjoon Lee, Bill Lin and Jia Wang, "The Taming of The Shrew: Mitigating Low-Rate TCP-Targeted Attack", *IEEE Transactions on Network and Service Management (TNSM)*, Vol. 7, no. 1, pages 1-13, March 2010.

Guanyao Huang, Chia-Wei Chang, Chen-Nee Chuah and Bill Lin, "Measurement-aware Monitor Placement and Routing: A Joint Optimization Approach for Network-Wide Measurements", *IEEE Transactions on Network and Service Management (TNSM)*, accepted to appear in 2011.

Chia-Wei Chang, Guanyao Huang, Bill Lin and Chen-Nee Chuah, "LEISURE: A Framework for Load-Balanced Network-Wide Traffic Measurements", *IEEE Transactions on Network and Service Management (TNSM)*, in submission.

Chia-Wei Chang, Han Liu, Guanyao Huang, Bill Lin, and Chen-Nee Chuah, "Distributed Measurement-Aware Routing for Multiple Classes of Flows", *IEEE Transactions on Networking (ToN)*, in submission.

ABSTRACT OF THE DISSERTATION

Efficient Gain-Driven Routing-Assisted Mechanisms for Network-wide Traffic Monitoring

by

Chia-Wei Chang

Doctor of Philosophy in Electrical Engineering (Communication Theory and Systems)

University of California, San Diego, 2011

Professor Bill Lin, Chair

Network-wide traffic monitoring is of interest to network operators. With constantly changing traffic characteristics and measurement objectives, existing techniques for traffic monitoring tend to be sub-optimal due to poor choice of monitor deployment locations. Routing-assisted network monitoring mechanisms have successfully catered to these needs and are able to maximize the overall traffic monitoring utility of the network by strategically re-directing selected traffic sub-populations over existing deployed monitoring devices. Both the traffic measurement gain of the network and the load-balancing of measurement workloads across distributed monitoring devices are important performance metrics in the design of efficient routing-assisted traffic monitoring mechanisms. This thesis focuses on the design of gain-

driven routing-assisted monitoring mechanisms where maximizing the overall traffic measurement gain is our primary design objective. This problem is tackled using two different approaches. First, novel centralized optimal and heuristic routing solutions are proposed for jointly optimizing monitor placement and dynamic routing strategy to achieve maximum measurement gain of the network. Next, we consider the load-balancing problem about how to distribute the network measurement workload across monitoring devices without compromising on the overall traffic measurement gain of the network. Providing effective load-balancing is important since previously-placed monitoring devices may be easily overwhelmed with ever-increasing link rates and increasingly sophisticated measurement tasks. We present an optimization framework called LEISURE (Load- EqualIzed meaSUREment) for load-balancing network measurement workloads across distributed monitors. Finally, a distributed measurement-aware traffic engineering protocol is proposed based on a game-theoretic re-routing policy that attempts to optimally utilize existing monitor locations for maximizing the traffic measurement gain of the network while ensuring that the traffic load distribution across the network satisfies some traffic engineering constraint. It guarantees not only a provable Nash equilibrium, but also a quick convergence without significant oscillations to an equilibrium state in which the measurement utility of the network is close to the maximum achievable gain using offline, centralized routing-assisted network monitoring mechanisms. Both these centralized and distributed routing-assisted approaches improve the overall traffic measurement utility of the network significantly while ensuring low computation complexity.

Chapter 1

Introduction

1.1 Network-Wide Traffic Monitoring

Comprehensive traffic monitoring is essential to a variety of network management tasks, including traffic engineering (TE), assessing performance, capacity planning, accounting, anomaly detection, and security forensics. Many existing studies focus on the design of improving traffic measurement techniques at a single monitor, including adaptive sampling [61], data streaming [47], and heavy-hitter detection mechanisms [30]. These solutions typically examine packet headers to determine if any statistics need to be collected. While these aggregate traffic volume statistics are sufficient for TE purposes, there is an increasing need for fine-grained flow level measurements to perform accurate traffic classifications for security purposes. For example, deep packet inspection (DPI) allows post-mortem analysis of network events and helps understand the payload properties of transiting Internet traffic. Network DVR [17], a programmable application-aware triggered trace collection system, performs precisely the function of packet content recording based on user-specified signatures. This in turn significantly reduces the number of memory copies for valid trace collection,

But doing such fine-grained flow level measurements to analyze packet payload is often an expensive process that requires dedicated hardware (e.g., TCAMs [74]), specialized algorithms, (e.g., Bloom Filters [25]), or vast storage capacity. Given the fast-changing Internet traffic landscape and large traffic volume, a single monitor is not capable of accomplishing the measurement tasks from all applications of interest due

to its resource constraint. This calls for coordinated measurement between multiple distributed monitors. Recent work has demonstrated the benefits of a network-wide coordinated measurement for traffic engineering [31, 76] and network diagnosis [48, 49, 51]. CSAMP, a centralized coordinated measurement proposal in [66], can significantly reduce management complexity and operating costs as [10, 14, 36]. Specifically, network-wide traffic measurements provide essential data for network operation and research. For example the strategy to obtain network information through end-to-end measurements, known as Internet tomography (e.g., mainly for topology discovery, or link delay monitoring), is therefore of great interest to the research community [35, 45, 70]. Moreover, network-wide traffic measurement at multiple monitors is also key to uncovering global network behavior since a single monitor only provides partial views and may not be sufficient or accurate. For example, a *global iceberg* [40] may have high aggregate volume across many different monitors, but may not be detectable at any single monitor. Discovering this type of event is important for a number of applications (e.g. detecting DDoS attacks, discovering worms, as well as ensuring SLA compliance).

1.2 Motivation for gain-driven network monitoring mechanisms design

This thesis focuses on the design of delivering high traffic measurement gain/utility in network-wide traffic monitoring. In this section, we discuss the motivation behind solving this problem. Network-wide traffic measurement can be classified as two categories: passive monitoring and active monitoring. The passive approaches deploy monitoring devices to the links in order to monitor the traffic which passes through the network while the active approaches generate explicit control messages/packets in the network for various measurement tasks. Active network monitoring has received much more attention than passive monitoring in the literature. Recent researches show that active network monitoring can be used to locate failures in IP networks [12, 38, 58]. In fact, IP networks do not typically generate feedback packets or state information and it is the reason why active network monitoring is needed to perform traffic engineering. Usually, active network monitoring implies additional overhead traffic and therefore its

objectives are to (1) find the minimum number of beacons which are used for emitting probe packets to cover all the links in the network as [12, 38] and (2) compute the smallest set of probe packets which has to send after the minimum number of beacons are chosen. Bejerano et al. [12] show that this problem is NP-complete. Nguyen et al. propose a different approach to solve this problem by starting from a set of possible beacons in [58]. They reversely first compute an optimal set of probe packets and then try to find the minimal number of beacons which are needed to generate these probe packets . They show that this beacon placement problem is also NP-hard and propose their greedy algorithm to solve it: in each iteration, they select a beacon, remove its corresponding set of probe packets (e.g., sent by this beacon) until the optimal probe packets set cannot be covered.

On the other hand, passive network monitoring is the basis for network operators to provide the robust, efficient, and secure operation of modern computer networks. Traditionally, passive network monitoring has been used for relatively simple traffic measurement task (e.g., gathering packet traces) for off-line analysis. Now it becomes vital for a wide class of more CPU and memory intensive network measurement applications, such as accurate traffic categorization [8], resource provisioning, network dimensioning, Traffic Dispersion Graph (TDG) analysis [42] and Network Intrusion Detection Systems (NIDS) [56, 64]. Chang et al. in [18, 19] present a simple priority-tagging filtering mechanism, called SAP (Shrew Attack Protection), that protects well-behaved TCP flows against low-rate TCP-targeted Shrew attacks. In this passive network monitoring scheme, routers maintain a simple set of counters and keeps track of the drop rate for each potential victim. If the monitored drop rates are low, all packets are treated as normal (e.g., low-priority) and equally compete to be admitted to the output queue and only dropped based on the AQM (Active Queue Management) policy when the output queue is (nearly) full. However, if the drop rate for a certain victim becomes higher than some dynamically determined threshold (called fair drop rate), the router treats packets for this victim as high-priority, and these high-priority packets are preferentially admitted to the output queue. SAP keeps tagging victim packets as high priority until their drop rate is below the fair drop rate. By preferentially dropping normal packets to protect high-priority packets, SAP can prevent low-rate TCP-targeted

Shrew attacks from causing a well-behaved TCP flow to lose multiple consecutive packets repeatedly. This simple strategy protects well-behaved TCP flows away from near zero throughput (due to slow start) under an attack. However, this passive traffic protection mechanism consume more CPU and memory overheads.

Network DVR, a novel passive traffic monitoring system, is a programmable application-aware triggered trace collection system which is proposed in [17]. It performs precisely the function of packet content recording based on user-specified trigger signatures. This in turn significantly reduces the number of memory copies that the passive monitoring system has to consume for valid trace collection, which has been shown previously as a key indicator of system performance [23]. However it introduces additional deployment and operation costs (e.g., CPU and memory usage). Another new passive monitoring methodology in [44] is to keep tracking both TCP's congestion window of the senders and their round trip time (RTT) in order to provide a valuable diagnostic of end-user-perceived network performance. It places passive devices to monitor the traffic on the link and collect the useful parts of packets with their arrival time-stamps which also increases more operation cost. Mainly, network operators have to deploy/operate specific tools or devices to monitor the network traffic passively. In contrast to active network monitoring, passive network monitoring does not introduce additional traffic overhead in the network. Unfortunately, the devices deployed in the links which monitor the network traffic usually introduce expensive deployment/operation cost due to the requirements for processing packets and storing collected measured data. Therefore the main objective of passive network monitoring is to minimize these costs (e.g., simply imply the number of monitoring devices) to cover all of the targeted traffic in the network.

In all of the network monitoring methodologies listed above, their key common objective is to minimize their measurement overheads, in terms of the number of active beacons and volume of additional probe packets for active network monitoring or in terms of the deployment cost, as well as operation/management cost for passive network monitoring. However with the fast-changing Internet traffic landscape and large traffic volume, monitors are not capable of accomplishing the measurement tasks from all applications of interest and monitoring all the targeted traffic in the network.

Also, collecting traffic data and analyzing such data from a Tier-1 backbone core network is real challenging since it is time-consuming and expensive to deploy passive monitoring/recording devices or active beacons in operational network. Moreover, the range of traffic volume on the links is from 10 Mb/s on OC-3 to 10 Gb/s on OC-192 backbone links, which means the monitoring devices involve processing terabytes of data. Sampling now is crucial since monitoring devices are not able to sustain a 100% traffic measurement coverage on high speed links (e.g., OC-48, OC-192 or higher) due to their resource constraints (e.g., the processing speed, memory storage). Therefore how to maximize the traffic measurement gain of the network or traffic monitoring coverage within limited exploitation overheads (e.g., deployment and operation costs) becomes an important issue in recent designs for network-wide traffic monitoring mechanisms.

1.3 Techniques for improving the traffic measurement gain in network-wide traffic monitoring

Improving the traffic measurement gain in network-wide traffic monitoring is a hard problem that has attracted significant interest in the literature. Several solutions have been proposed for different contexts and their methodologies are categorized as (1) deriving better **monitor placement** strategies across the network, (2) finding both better **monitor placement** strategies and proper **configuration decisions** of monitors, (3) using disjoint **flow sampling** and (4) deriving better **routing strategies** for different traffic sub-populations.

1. **Monitor placement:** Early work on network-wide traffic monitoring has focused on the placement of monitors at appropriate locations to cover all routing paths using as few monitors as possible [12, 20, 38, 45, 58, 72]. In [45], the authors focus on the placement of measurement devices for active traffic monitoring, specifically for the construction of distance maps while in [12, 38, 58], they address the placement problem in an active monitoring infrastructure to measure delays and detect link failures. Chaudet et al. in [20] study the problem of minimizing the number of monitoring devices for passive traffic monitoring and

finding optimal strategic locations of beacons for active traffic monitoring. They also present a combinatorial view of the problem, giving rise to approximability and complexity results, as well as efficient and versatile Mixed Integer Programming (MIP) formulations. Several greedy solutions are proposed by using this modeling. Moreover, from this new model, they are able to derive MIP even for the minimization of the deployment and the exploitation cost when maximizing the total traffic measurement gain of the network. Suh et al. in [72] present heuristics for placing passive monitoring devices in POP where each of these devices only captures a portion of the traffic carried by the link. They also consider how to maximize the volume of captured traffic under resource constraints of the monitoring devices where each of them has its own deployment and operational costs.

2. **Monitor placement and corresponding configuration decisions:** There are extensions to the monitor-placement problem in [72] to incorporate with configuration decisions of these monitors (e.g., packet sampling rates). They consider the problem of where to place monitoring devices in the network and how to control their sampling rates. To address the tradeoff between measurement overheads (e.g., deployment/operation costs) and traffic monitoring coverage, they consider both minimum cost and maximum coverage problems under various budget constraints. Specifically, they consider three main problems: (1) minimizing the deployment cost of monitoring devices to achieve a monitoring objective/task, (2) minimizing both deployment and operation costs of monitoring devices under the same objective/task and (3) maximizing the fraction of IP flows being sampled (e.g., the traffic measurement gain) by addressing the problem of placing monitors in proper locations and setting their corresponding sampling rates. However they show that all of these defined problems are NP-hard. They further propose a two phase greedy heuristic approach to maximize the traffic measurement gain where they first find the links that should be deployed monitoring devices and then run a second optimization algorithm to adjust their sampling rates. They show that this two-phase heuristic provides solution quite close to the optimal one through experiments using synthetic and real network topologies discovered by the

Rocketfuel utility and with generated traffic matrices. Cantieni et al. also consider a similar problem in [15] but they reformulate the monitor placement problem as follows: given a network where all of the links are deployed monitoring devices, how to decide which devices should be activated and what sampling rate should be adjusted on these monitoring devices in order to achieve a given measurement task with high accuracy (e.g., maximize the traffic measurement gain) and low resource consumption? In contrast to [72], their optimization framework can solve both the selection of activated monitors and the configurations of their packet sampling rates in one step.

3. **Flow sampling:** As we mentioned before, modern traffic monitoring devices cannot each record all packets of interest or flows that pass through due to their technological and resource constraints (e.g., CPU and memory resources). Coupled with ever-increasing link rates (high traffic volume), they rely on a variety of sampling techniques to selectively record as many packets as their CPU and memory resources allow. While sampling makes passive measurement technologically feasible (i.e., operate within the resource constraints of monitoring resources), the overall fidelity of flow-level measurements is reduced. The reason is because in today's networks, the monitoring devices record flow measurements completely independent to each other, thus leading to redundant measurements and inefficient use of device resources. Sekar et al. [66] show that a centralized system, Coordinated Sampling (CSAMP), that coordinates disjoint monitoring responsibilities across different monitoring devices can significantly improve the flow monitoring capabilities (e.g., traffic coverage) of a network. Instead of using traditional packet sampling, CSAMP uses not only flow sampling (e.g., proposed by by Hohn et al. in [37]) to avoid the sampling biases against small flows but also a hash-based packet selection as a router-level primitive (e.g., Trajectory Sampling in [26]) to eliminate duplicate flow measurements in the network to improve the traffic measurement gain of the network. This allows multiple monitoring devices to measure disjoint sets of flows without requiring explicit communication between routers, thus eliminating redundant and possibly ambiguous traffic measurements across the network. By using this disjoint flow

sampling model, CSAMP formulates an optimization framework to specify its network-wide monitoring objective as maximizing the total flow-coverage across all OD-pairs traffic subject to ensuring that the minimum fractional coverage per OD-pair can be achieved while respecting the resource constraints of monitoring device. The output of this optimization is then translated into per-monitor sampling manifests that specify the set of flows that each monitoring device is required to measure. They evaluate the benefits of CSAMP over a wide range of network topologies and show that it can observe more than twice as many flows compared with traditional uniform packet sampling. Also it is more effective to achieve network-wide traffic monitoring goals.

4. **Routing strategies:** Several past research efforts have focused on the optimal deployment of monitoring devices in operational networks to improve the traffic measurement gain. Such deployment involves both monitoring device placement as well as configuration decisions (e.g., packet sampling rates). The optimal placement and configuration of monitoring devices for a specific measurement task typically assumes a priori knowledge about the traffic characteristics. Moreover, these priori knowledge are typically performed at long-time scales to allow provisioning of required physical resources. However, traffic characteristics and measurement objectives may change dynamically, potentially rendering a previously determined optimal placement suboptimal. It is not feasible to dynamically redeploy/reconfigure monitoring devices in the network infrastructure to cater such evolving measurement requirements. Raza et al. propose a routing-assisted framework called MeasuRouting in [63] to address this problem by strategically re-directing traffic sub-populations of interest over existing deployed monitoring devices to maximize the traffic measurement gain of the network. MeasuRouting takes deployment locations of monitoring devices as an input and only decides how to route network traffic/flows. Since routing decision for every packet is made dynamically at every router, MeasuRouting can conceptually adjust those routing decisions to both evolving traffic patterns and different measurement tasks given by network operator to maximize the overall monitoring utility of the network where the overall monitoring utility is defined as a weighted sum of the

traffic amount measured over all flows. The main challenge for routing-assisted traffic monitoring mechanisms are to work within the constraints of existing intra-domain traffic engineering (e.g., bandwidth resources, or Quality of Service (QoS) constraints). In general, intra-domain routing is often specified for aggregate flows. MeasuRouting, can therefore, differentially route traffic sub-populations of an aggregate flows while ensuring that the aggregate placement is compliant to original traffic engineering objectives. They define three classes of traffic engineering objectives in [63] for routing-assisted traffic monitoring mechanism, each differing in the level of required conformity to the original routing: 1) Least TE Disruption MeasuRouting (LTD): The basic version of routing-assisted traffic monitoring (MeasuRouting) problem and it requires only that the aggregate TE policy is not violated. 2) No Routing Loops MeasuRouting (NRL): NRL is proposed to ensure that the micro-flowset[1] routing is loop-free since the flow conservation constraints in LTD do not guarantee the absence of loops. However, depending upon the exact forwarding mechanisms and routing protocol, NRL may still not be feasible. 3) Relaxed Sticky Routes MeasuRouting (RSR): RSR ensures that the new micro-flowset routing does not pass through a link that the macro-flowset[2] traffic was not routed before in the original routing. It means RSR guarantees feasible micro-flowset routing.

The work presented in this thesis focuses on the design of using routing-assisted mechanisms and monitor placement strategies as primary means to improve the traffic measurement gain in network-wide traffic monitoring. The configuration decisions of the monitoring devices are assumed unadjustable and given (e.g., packet sampling rates) for all traffic flows. Disjoint flow sampling can be used in conjunction with the ideas proposed in this thesis to further improve the efficiency (e.g., eliminate measurement redundancy) and monitoring ability (e.g., the traffic measurement gain) of the system.

[1] A macro-flowset (e.g., per origin-destination (OD)-pair traffic) may consist of multiple micro-flowsets.

[2] A macro-flowset represents a set of flows for which an aggregate routing placement is given and has the same ingress and egress nodes.

1.4 Problem statement and contributions

The problem solved in this thesis can be formally stated as follows:

How can we design efficient routing-assisted mechanisms that deliver the high traffic measurement gain for network-wide traffic monitoring while ensuring low deployment cost, low operation cost and adhere to the intra-domain traffic engineering constraints ?

In this thesis, the above problem is solved using three of the four different approaches discussed in Section 1.3. First, novel centralized optimal and heuristic routing solutions are proposed for jointly optimizing monitor placement and dynamic routing strategy to achieve maximum measurement utility of the network where traffic characteristics and monitor capacities are given as inputs. In particular, all proposed heuristic routing algorithms can approach the maximum measurement utility of the optimal one but yet they require dramatically shorter computation times. Second, in addition to their traffic measurement gain optimality, we further present a load-balanced optimization framework to distribute the network measurement workload across participated monitoring devices without compromising on the overall traffic measurement gain of the network by using disjoint flow sampling. Specifically, we consider various load-balancing problems under different optimization objectives and study their extensions to support more realistic scenarios. Next, a distributed measurement-aware traffic engineering protocol is proposed based on a game-theoretic re-routing policy that attempts to optimally utilize existing monitor locations for maximizing the traffic measurement gain of the network while ensuring that the traffic load distribution across the network satisfies some traffic engineering constraint.
The main contributions of this thesis are as follows:

- **Achieving Maximum Measurement Utility of the Network by Jointly Optimizing Monitor Placement and Dynamic Routing Strategy:** A new Measurement-aware Monitor Placement and Routing framework (MMPR) that jointly optimizes monitor placement and dynamic routing strategy is proposed to achieve maximum measurement utility of the network where the overall

measurement utility is quantified as how well each individual flow is monitored (e.g., how many bytes or packets are sampled), weighted by its importance. The main challenge of MMPR is to decouple the relevant decision variables and adhere to the intra-domain traffic engineering constraints. We formulate the MMPR problem as an MIQP (Mixed Integer Quadratic Programming) problem, and show how it could be reformulated as a standard MILP (Mixed Integer Linear Programming) problem by decoupling the two key decision variables (e.g., monitor-placement and traffic-routing decision variables). In our framework, the optimal routing strategy is determined for each flowset, which is defined to be any aggregation of flows sharing the same ingress/egress routers and having the same routing decision. We strive to adhere to the existing intra-domain traffic engineering (TE) constraints such that we maintain similar maximum link utilization in the network as in default routing case. We also attempt to constrain measurement resources by activating no more than K monitors in arbitrary links. We investigate several approximate solutions that can approach the performance of the optimal MILP solution, but yet they require dramatically shorter computation times. Our heuristic algorithms include K-Best, Successive Selection, Greedy and Quasi-Greedy. We perform detailed simulation studies using real traces and topologies from Abilene [1], AS6461 [70], and GEANT [3]. Our results show that the optimal MMPR solution can achieve measurement gains up to a factor 1.76X better when compared to baseline cases (i.e., optimal Placement-only or MR(MeasuRouting)-only). We also show that our heuristic algorithms can achieve measurement utilities that are quite close to the optimal solution, while reducing computation times by a factor of 23X in Abilene, 246X in AS6461, and 233X in GEANT, compared with the MILP (optimal) solution. The details of this work are discussed in Chapter 2.

- **Presenting Load-Balanced Network-Wide Traffic Measurement without Compromising on the Overall Maximum Measurement Utility of the Network:** A new centralized optimization framework called LEISURE (Load-EqualIzed meaSUREment) is proposed to address the network measurement load-balancing problem on various realistic scenarios while ensuring that the

maximum measurement utility of the network is achieved. LEISURE distributes traffic measurement tasks evenly across coordinated monitoring devices subject to ensuring that the required fractional coverage of those tasks (e.g., given from MMPR) can be achieved. It takes a) routing matrix, b) the topology and monitoring infrastructure deployment and c) measurement requirements of tasks as inputs, and decides which available monitoring devices should participate in each specific measurement task and how much they need to measure to optimize the load-balancing objectives. The load-balancing objective in this thesis is mainly defined as two terms: 1) minimizing the variance of workloads across all monitors or 2) minimizing the maximum workload among them. The optimal outputs/solutions are translated into the disjoint sets of required-measured flows that each monitor is assigned to measure. We also propose simple heuristic solutions to compare with the optimal one and extend LEISURE to incorporate practical scenarios (constraints), i.e., (a) with limited measuring resources at monitors, (b) with limited number of deployed monitors, (c) with multiple routing paths (e.g., ECMP) for each origin-destination (OD)-pair traffic. As proof of concept, we perform detailed simulation studies based on Abilene [1] and GEANT [3] network topologies and traces. Our results show that the significant load-balancing improvement (e.g., 4.75X smaller maximum workload and 70X smaller variance in workloads) is achieved by using LEISURE to optimally distribute the measurement tasks across all coordinated monitors when compared with the naive uniform assignments. We also present detailed performance comparison of our proposed heuristic algorithms belonging to two categories: LB-Greedy and LB-Successive Selection in flexible monitor deployment scenario. We show that our proposed heuristic solutions can achieve load-balancing performance that are quite close to the optimal solutions, while reducing the computation times by a factor up to 22.5X in Abilene and 800X in GEANT. We extend LEISURE and simulation studies to perform optimizations and sensitivity analysis with respect to multiple measurement tasks that exhibit different importance and incur different costs. We show that LEISURE is flexible enough to assign the correct set of measurement tasks for coordinated monitors

to optimize measurement utility given limited measuring resources. The design, implementation, and evaluation of LEISURE is described in Chapter 3.

- **Developing Distributed Measurement-Aware Traffic Engineering Protocol to Achieve Maximum Measurement Utility of the Network:** A distributed measurement-aware traffic engineering protocol, Distributed MeasuRouting (DisMR), is proposed based on a game-theoretic re-routing policy that attempts to optimally utilize existing monitor locations for maximizing the traffic measurement gain of the network while ensuring that the traffic load distribution across the network satisfies some traffic engineering constraints. DisMR takes advantage of alternative paths in a network (e.g., equal cost multi-path routing (ECMP)). It maximizes the traffic measurement gain by adjusting the traffic split ratios among these paths to the same destination. It actually operates on top of an existing multiple-path routing infrastructure (e.g., ECMP). DisMR is derived from a game-theoretic re-routing policy that captures the dynamic decision-making process and interactions among distributed routers. We introduce a novel cost function on each link that reflects both the measurement capabilities (gain) and the traffic engineering (TE) constraint (i.e., links with larger measurement resources have a smaller cost but links with a larger TE score (e.g., link utilization) have a larger cost). The cost function is designed such that flows are attracted to links with better measurement capabilities while avoiding TE violations. Routers compete with each other in a game-theoretic manner in order to minimize their own costs for the downstream paths. In DisMR, each router periodically gathers/propagates its sub-path cost information for upstream routers and use it to locally decide how to adjust traffic split ratios for each destination to the next-hop routers among these multiple equal-cost paths. Our routing policy guarantees not only a provable Nash equilibrium, but also a quick convergence without significant oscillations to an equilibrium state in which the measurement gain of the network is close to the maximum achievable gain using offline, centralized MeasuRouting. We evaluate DisMR via simulations using both synthetic and real traces/topologies from Abilene [1], AS6461 [70], and GEANT [3]. The simulation results show fast convergence (as expected from the

theoretical results), improved measurement gains (e.g., 12 % higher) and much lower TE-violations (e.g., up to 100X smaller) compared to static, centralized MeasuRouting in dynamic traffic scenario. The DisMR measurement-aware traffic engineering protocol is described in Chapter 4.

Chapter 2

Measurement-aware Monitor Placement and Routing: A Joint Optimization Approach for Network-Wide Measurements

2.1 Introduction

Given the sheer size and complexity of the Internet today and its increasingly important role in modern-day society, there is a growing need for high-quality network traffic measurements to better understand and manage the network. Obtaining accurate network-wide traffic measurement in an efficient manner is a daunting task given the multi-faceted challenges. First, there is an inherent lack of fine-grained measurement capabilities in the Internet architecture. Second, the rapidly increasing link speeds make it impossible for every router to capture, process, and share detailed packet information. Earlier work on traffic monitoring has focused on improving single-point measurement techniques, such as sampling approaches [22, 37], estimation of heavy-hitters [30], and methods to channel monitoring resources on traffic sub-populations [62, 75]. To achieve network-wide coverage, previous studies have focused on the optimal deployment of monitors across the network to maximize the monitoring utility (as determined by the

network operator) with given traffic routing [15, 20, 72]. The optimal placement for a specific measurement objective typically assumes a priori knowledge about the traffic characteristics. However, both traffic characteristics and measurement objectives can dynamically change over time, potentially rendering a previously optimal placement of monitors suboptimal. For instance, a flow of interest can avoid detection by not traversing the deployed monitoring boxes. The optimal monitor deployment for one measurement task might become suboptimal once the objective changes.

To address the limitation mentioned above, MeasuRouting [63] was recently proposed to strategically/dynamically route important traffic over fixed monitors such that it could be best measured. Using intelligent routing, it can cope with the changes of traffic patterns or measurement objectives to maximize measurement utility while meeting existing intra-domain traffic engineering (TE) constraints, e.g., achieving even load distribution across the network, or meeting Quality of Service (QoS) constraints. It is oblivious of the monitor placement problem. The key idea is that the routes of important and unimportant flows can be exchanged to achieve better measurement and load balancing. However, MeasuRouting is based on the assumption that monitor locations have already been decided a priori and fixed. It does not consider the flexibility of deploying new monitors and replacing old ones, or altering the existing monitor placement strategies.

In practice, current routers deployed in operational networks are already equipped with monitoring capabilities (e.g., Netflow [7], Openflow [59]). Network operators would not turn on all these functionalities because of their associated expensive operation cost [15, 20, 72] and measurement redundancy [66], and hence there are potentially hundreds of monitoring points to choose from to achieve network-wide measurements. Given routing could be changed dynamically to aid measurement, the optimal monitor selection/placement strategies may also change to take advantage of this new degree of freedom. Therefore, previous approaches that treat monitor placement and routing as two separate problems may be sub-optimal (as demonstrated in Section 2.2 with an example scenario). This naturally leads to the following open question: *Given a network where all links can be monitored, which monitors should be activated and how to strategically route traffic sub-populations over those planned*

monitors such that both the measurement gain is maximized and the limited resources is best utilized.

In this chapter, we propose an MMPR (Measurement-aware Monitor Placement and Routing) framework that jointly optimizes monitor placement and traffic routing strategy, given traffic characteristics and monitor capacities as inputs. In our framework, the optimal routing strategy is determined for each flowset, which is defined to be any aggregation of flows which share the same ingress/egress routers and have the same routing decision. The goal is to maximize the overall measurement utility, which quantifies how well each individual flow is monitored (e.g., how many bytes or packets are sampled), weighted by its importance. We strive to adhere to the existing intra-domain traffic engineering (TE) constraints such that we maintain similar load distributions in the network (e.g., maximum link utilization) as in default routing case. We also attempt to constrain measurement resources by activating no more than K monitors in arbitrary links.

The properties of monitors and importance of flows in this chapter are modeled in a very generic form such that our framework can be applied to a wide variety of measurement scenarios. We assume that the dynamic traffic/measurement changes will stay for long enough time for us to re-optimize monitor placement and flowset routing. Implementation issues for continuous measurement are discussed in Section 2.7 or left as future work. We highlight our contributions as follows:

- We formulate the MMPR problem as an MIQP (Mixed Integer Quadratic Programming) problem, and show how it could be reformulated as a standard MILP (Mixed Integer Linear Programming) problem by decoupling the two key decision variables.
- We investigate several approximate solutions that can approach the performance of the optimal MILP solution, but yet they require dramatically shorter computation times. Our heuristic algorithms include K-Best, Successive Selection, Greedy and Quasi-Greedy.
- We perform detailed simulation studies using real traces and topologies from Abilene [1], AS6461 [69], and GEANT [3]. Our results show that the optimal MMPR solution can achieve measurement gains up to a factor 1.76X better when compared to baseline cases (i.e., optimal Placement-only or MR(MeasuRouting)-

only). We also show that our heuristic algorithms can achieve measurement utilities that are quite close to the optimal solution, while reducing computation times by a factor of 23X in Abilene, 246X in AS6461, and 233X in GEANT, compared with the MILP (optimal) solution.

The rest of this chapter is organized as follows. Section 2.2 illustrates through a motivating example the benefits of a joint optimization approach that considers both monitor placement and traffic routing together. Section 2.3 formulates the MMPR problem, and Section 2.4 presents our heuristic solutions. Section 2.5 presents detailed experimental results using our proposed methods, and Section 2.6 outlines related work. Finally, Section 2.7 discusses practical implementation issues and concludes this chapter.

2.2 Motivating Example

In this section, we showcase the importance of both monitor placement and traffic routing through an illustration. Consider the topology in Figure 2.1. We define a flow based on the five tuple $< srcip, dstip, srcpt, dstpt, proto >$. We assume that due to budget considerations, only one monitor is allowed to be deployed in any one of the 12 links. The network operator wants to identify both the best monitor deployment location and the best routing strategy for “important" flows, to achieve maximum measurement gain, i.e., measuring as many important flows as possible. At the same time, the operator wants to ensure that the monitor placement and any routing changes have least impact on existing QoS metric, which is defined as the “average path length" of every flow.

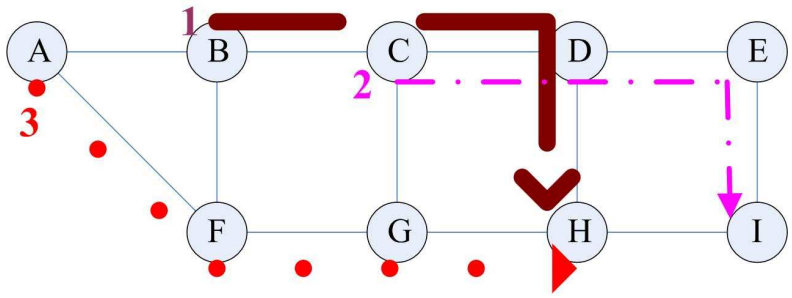


Figure 2.1: MMPR Motivational Example

Initially there are two important flows, flow 1 and 2, with their default routing show in Figure 2.1. Obviously the optimal monitor deployment location is on link $C \rightarrow D$ where the two important flows traverse. There are many other unimportant flows (not shown in the figure) from each OD (origin-destination) pair. All of the N unimportant flows (including flow 3) use shortest path routing. Suppose their average path length is ζ.

Suppose now flow 3 becomes important over time, with its default route $A \rightarrow F \rightarrow G \rightarrow H$. With the current monitor placement (previously determined to be optimal), flow 3 will not be monitored at all. In order to capture this flow, a second monitor (additional resources) will be needed along the path $A \rightarrow F \rightarrow G \rightarrow H$ if the routing remain unchanged. Alternatively, a dynamic routing approach like MeasuRouting would redirect flow 3 through link $C \rightarrow D$ (assuming the resulting link utilization is below a desired threshold). However, this detour increases path length for flow 3 from 3 to 4. Since every other flow uses shortest path routing, the average path length increases from $\frac{N\zeta+6}{N+2}$ to $\frac{N\zeta+7}{N+2}$, which clearly has a negative impact on the QoS metric.

Instead, it would be better to move the monitor from $C \rightarrow D$ to link $G \rightarrow H$, and redirect the flow 1 and 2 both through link $G \rightarrow H$. The new routes for flow 1 and 2 can be $B \rightarrow F \rightarrow G \rightarrow H$ and $C \rightarrow G \rightarrow H \rightarrow I$, respectively. As such, no flow has increased its path length, i.e. average path length remains $\frac{N\zeta+6}{N+2}$. All flows can be monitored with only one monitor (without additional resources).

One other practical concern is that the redirection of flow 1 and 2 may overload link $G \rightarrow H$. This can be simply avoided by switching flow 2 with another unimportant flow from B to H, as long as that flow has equal traffic amount and was originally routed through $B \rightarrow F \rightarrow G \rightarrow H$. Flow 3 can be similarly treated by switching with a flow originally routed as $C \rightarrow G \rightarrow H \rightarrow I$. MeasuRouting [63] has already shown ways to switch flows for better measurement. In our situation, by switching flow 1 and 2 with other unimportant flows, both average path length and link load can be preserved at initial conditions. The same scenario may lead to different optimal solutions (the new placement location and new routes) with other TE metric definitions. The problem becomes more complicated with more important flows, larger topology and different TE

metrics.

The example above reveals that MeasuRouting without considering changing monitor placement (referred to as MeasuRouting- or MR-only) may become suboptimal. Similarly, changing the optimal monitor placement alone (referred to as Placement-only) without the flexibility in re-routing may be infeasible without introducing additional measurement resources (e.g., adding a second monitor in this example). A better monitor placement combined with strategic routing can achieve optimal solution (maximum measurement gain) while meeting both the QoS or TE constraints. This motivates us to formulate the joint optimization problem of both monitor placement and traffic routing under the MMPR framework and propose optimal solutions that achieve best measurement utility with limited monitor resources. We will later compare the performance of optimal MMPR solution with MR-only and Placement-only in Figure 2.2. In the example above, Placement-only strategy will miss flow 3 completely. Both MR-only and MMPR can monitor all flows. However, MR-only increases the average path length (QoS metric) to $\frac{N\zeta+7}{N+2}$, which is undesirable, while MMPR reduces it to $\frac{N\zeta+6}{N+2}$.

The main focus on this chapter is to provide a *theoretical* framework for MMPR problem and examine the cost/performance trade-offs for the optimal solution and a variety of heuristic approaches. There are several practical issues which remain to be addressed in order to realize MMPR solutions. For example, MMPR assumes prior knowledge of traffic importance, which is usually inaccurate in practice. All the related implementation issues will be discussed in Section 2.7.

2.3 MMPR FRAMEWORK

We now present a formal framework for MMPR in the context of a centralized architecture, which jointly optimizes monitor placement and traffic routing assuming it has global knowledge of $a)$ the network topology, $b)$ the size and importance of traffic sub-populations, $c)$ the monitor capability, and $d)$ the TE policy.

2.3.1 Definition

$G(V, E)$ represents our network, where V is the set of nodes and E is the set of directed links. $M = |V|$ is the total number of links. An OD pair represents a set of flows between the same pair of ingress/egress nodes for which an aggregated routing placement is given. The set of all $|V| \times |V - 1|$ OD pairs is given by Θ. Γ_{ij}^{x} denotes the fraction ($[0, 1]$) of the traffic demand belonging to OD pair x placed along link (i, j). $\{\Gamma\}_{(i,j)\in E}^{x\in\Theta}$ is an input to the MMPR problem and represents our *original routing*. We assume $\{\Gamma\}_{(i,j)\in E}^{x\in\Theta}$ is a *valid* routing, i.e. flow conservation constraints are not violated and it is compliant with the network TE policy.

An OD pair may consist of multiple flows where some of them have higher measuring importance than others. The purpose of traffic measurement is to capture those important flows as much as possible. However, it is impractical to enforce individual routing decision for each flow. On the other hand, flows are aggregated as flowsets according to flow semantics, e.g: prefix based routing. In this paper, we define flowset to be any aggregation of flows which share the same ingress/egress routers and have the same routing decision. We use θ to denote the set of mutually exclusive flowsets and Υ_x to denote the set of flowsets that belongs to the OD pair x. Each flow is assigned to one flowset in θ.

We denote the fraction of traffic demand of flowset y placed along link (i,j) as γ_{ij}^{y}. $\{\gamma\}_{(i,j)\in E}^{y\in\theta}$ represents our flowset routing and is the set of decision variables of the MMPR problem. According to this definition, flows belonging to the same flowset y should have the same routing. We denote $\{\Phi\}_{x\in\Theta}$ and $\{\phi\}_{y\in\theta}$ to be the traffic demands (e.g., the sizes) for the OD pair Θ and flowset θ, respectively. It follows that $\Phi_x = \sum_{y\in\Upsilon_x} \phi_y$. $\mathcal{I}_{y\in\theta}$ denotes the measurement utility of the flowset y. This is a generic metric that defines the importance of measuring a flowset, which is related to the importance of its individual flows.

In this chapter, we assume traffic measurements are conducted on links. We define our measurement infrastructure and measurement requirement in abstract terms. $\{S\}_{(i,j)\in E}$ denotes the measurement characteristic of all links, i.e. the ability of a link to measure traffic. For example, $S_{(i,j)}$ can be equal to p_{ij}, the sampling rate of link (i, j). Since packet sampling is the de facto deployed measurement method, we will use

Table 2.1: Summarization of Notations

Notation	Description
x	OD pair
y	flowset
Θ	set of OD pairs
ϕ	set of flowsets
Φ_x	traffic demands of the OD pair x
ϕ_y	traffic demands of the flowset y
Γ_{ij}^x	original routing for OD pair x
$\mathcal{S}_{(i,j)}, p_{(i,j)}$	measurement characteristic of link (i,j)
$\mathcal{I}_y$	measurement utility of the flowset y
T_y	measurement gain of flowset y
$\gamma_{(i,j)}^y$	routing decision variable for flowset y
$u_{(i,j)}$	monitor placement variable for link (i,j)
β	optimization objective

p_{ij} and $\{S\}_{ij}$ interchangeably to denote the measurement ability of each link, and we discuss other possible measurement functions in Section 2.3.3. In summary, $\{S\}_{(i,j)\in E}$ and $\mathcal{I}_{y\in\theta}$ are inputs given to our MMPR problem.

Another input to the MMPR problem is K, the maximum number of monitors that would be turned on inside the network. In this paper, monitors can be turned on any of the M links. The (0,1) boolean variable u_{ij} is used to denote the placement strategy. Finally, we use a *measurement resolution function* (β) to characterize the overall performance of traffic measurement. β assigns a real number representing the monitoring effectiveness of flowset routing, flowset utility, and monitor placement strategy for given measurement characteristics. The objective of MMPR is to maximize β. Notations are summarized in Table 2.1.

$$\beta : (\{\gamma\}_{(i,j)}^y, \{S\}_{(i,j)}, \{\mathcal{I}\}_y, u_{(i,j)}) \rightarrow \Re \tag{2.1}$$

2.3.2 Formulation

In our problem, we can formulate the measurement gain through two kinds of popular reward models [72]. Let utility function T_y denote the benefit gained by

monitoring flowset y. We assume that there is no additional benefit gained by repeatedly monitoring the same traffic. Thus T_y can be expressed in either of two ways:

$$T_y = 1 - \prod_{(i,j)\in E} (1 - p_{ij} u_{ij} \gamma_{ij}^y) \tag{2.2}$$

$$T_y = \sum_{(i,j)\in E} p_{ij} u_{ij} \gamma_{ij}^y \tag{2.3}$$

Equation (2.3) approximates Equation (2.2) if $p_{ij} u_{ij} \gamma_{ij}^y$ is very small. This is true for most core-networks since the sheer traffic volume/speed prohibits high rate measurement. Equation (2.2) models the case where monitors independently sample flows, while in Equation (2.3), monitors measure non-overlapping traffic. This can be achieved by CSamp [66] likely methods, in which disjoint hash-based filters are placed before flows get sampled. In this chapter, we use the later reward model since it is linear, allowing us to better compare the various MMPR solutions.

$$\texttt{Maximize } \beta \tag{2.4}$$

$$\beta = \sum_{y\in\theta} \mathcal{I}_y T_y \tag{2.5}$$

$$= \sum_{y\in\theta} \sum_{(i,j)\in E} \mathcal{I}_y p_{ij} u_{ij} \gamma_{ij}^y \tag{2.6}$$

$$\gamma_{ij}^y \geq 0, \forall y \in \theta, (i,j) \in E \tag{2.7}$$

$$u_{ij} \in \{0,1\}, \forall (i,j) \in E \tag{2.8}$$

In our model, T_y is the summation of the product of p_{ij}, γ_{ij}^y, and u_{ij}. Therefore the objective function β is related to the product of two decision variables u_{ij} and γ_{ij}^y, and the optimization problem falls into the MIQP (Mix Integer Quadratic Programming) category. In order to avoid quadratic programming, we introduce z_{ij}^y to decouple $u_{ij} \times \gamma_{ij}^y$ by Equations (2.10) and (2.11). It is easy to see their equivalence. When $u_{ij} = 0$, $z_{ij}^y = 0$ from (2.10); and when $u_{ij} = 1$, $z_{ij}^y = \gamma_{ij}^y$ from (2.11).

$$z_{ij}^y = \gamma_{ij}^y \times u_{ij} \tag{2.9}$$

$$0 \leq z_{ij}^y \leq u_{ij} \tag{2.10}$$

$$\gamma_{ij}^y + u_{ij} - 1 \leq z_{ij}^y \leq \gamma_{ij}^y \tag{2.11}$$

After we substitute Equation (2.10)-(2.11) to Equation (2.6), the formulation becomes MILP (Mixed Integer Linear Programming) instead of MIQP:

$$\texttt{Maximize}\ \beta \tag{2.12}$$

$$\beta = \sum_{y\in\theta}\sum_{(i,j)\in E} \mathcal{I}_y p_{ij} z_{ij}^y \tag{2.13}$$

$$0 \le z_{ij}^y \le u_{ij} \tag{2.14}$$

$$\gamma_{ij}^y + u_{ij} - 1 \le z_{ij}^y \le \gamma_{ij}^y \tag{2.15}$$

$$\gamma_{ij}^y \ge 0, \forall y \in \theta, (i,j) \in E \tag{2.16}$$

$$u_{ij} \in \{0,1\}, \forall (i,j) \in E \tag{2.17}$$

We set the maximum number of allowed monitors to be no more than K:

$$\sum_{(i,j)\in E} u_{ij} \le K \tag{2.18}$$

After introducing MMPR, the new routing should not violate the TE metric (e.g., maximum link utilization) by more than a certain threshold, as compared with original routing. We use σ^Γ and σ^γ to denote TE metric of original routing and new routing, respectively. We introduce a threshold ϵ, which bounds the violation of TE metric.

$$\sigma^\gamma \le (1+\epsilon)\sigma^\Gamma \tag{2.19}$$

The traffic constraints can be formulated as follows:

$$\sum_{i:(i,j)\in E} \gamma_{ij}^y - \sum_{k:(j,k)\in E} \gamma_{jk}^y = 0 \qquad y \in \theta, j \ne in_y, out_y \tag{2.20}$$

$$\sum_{i:(i,j)\in E} \gamma_{ij}^y - \sum_{k:(j,k)\in E} \gamma_{jk}^y = -1 \qquad y \in \theta, j = in_y \tag{2.21}$$

$$\sum_{i:(i,j)\in E} \gamma_{ij}^y - \sum_{k:(j,k)\in E} \gamma_{jk}^y = 1 \qquad y \in \theta, j = out_y \tag{2.22}$$

In MMPR, to maximize β, important flowsets might get repeatedly routed through monitors. In reality, loop-free routing is desirable to avoid huge delays. MeasuRouting [63] proposed two methods (RSR and NRL) to provide candidate routes which are loop-free. They pre-calculate allowable acyclic paths for each OD pair. The

optimization problem then selects the best routes from these candidates. In this chapter, we borrow the idea of NRL (No Routing Loops MeasuRouting [63]). It allows us to select paths other than original routing Γ^x, by introducing $\Psi_{x:y\in\Upsilon_x}$:

$$\gamma_{ij}^{y} = 0 \qquad\qquad y \in \theta, (i,j) \notin \Psi_{x:y\in\Upsilon_x} \tag{2.23}$$

Equation (2.23) states that only links included in $\Psi_{x:y\in\Upsilon_x}$ may be used for routing flowset y. We use the heuristic algorithm in [63] to construct these paths. For each OD pair, it iteratively adds new link to $\Psi_{x:y\in\Upsilon_x}$ in decreasing order of sampling rate, as long as it does not introduce any loop.

2.3.3 Extensions

In this section, we extend our formulation and discuss some related issues. First, our formulation only introduces parameter K to bound the number of monitors, without formulating any detailed cost functions. In reality, the operation cost of monitors also depend on their sampling rates. Let f_{ij}(and g_{ij}) denote the unit monitor deployment (and operation) cost at link (i,j), and B(and C) represents the maximum budget for deployment (and operation) cost. We could add these two constraints as follows:

$$\sum_{(i,j)\in E} u_{ij} \times f_{ij} \leq B \tag{2.24}$$

$$\sum_{y\in\theta} \sum_{(i,j)\in E} p_{ij} u_{ij} \gamma_{ij}^{y} \times g_{ij} \times \phi_y \leq C \tag{2.25}$$

We can also treat the sampling rate, p_{ij}, as another decision variable if the operator tends to better configure the operation cost of monitors. The new problem becomes complicated since both the optimization objective (2.13) and the constraint (2.25) become quadratic. In reality, it is difficult to compare and tune the settings for different measurements. It is impractical to mathematically compare these costs with measurement gains. Instead, we formulate the fundamental situation where the cost is only related to the number of monitors, and each monitor has fixed configuration. It is equivalent to the case where f_{ij} is identical to all of the monitors.

Second, our formulation is based on “uniform" measurement. That means, each monitor will treat any traffic that traverse it equally. The objective function then

becomes linear. In reality, more sophisticated measurements can intelligently adapt to different flows [62, 75]. Because of this, the measurement gain function β might become nonlinear, or, other parameters are needed to reflect the difference in how flows are measured by the same monitor. We will explore different measurement methods (e.g.: flow sampling, flexsample [62], etc) in our future work. Our current formulation applies to any measurement scheme where all packets are treated equally by the same monitor. For example, DPI (deep packet inspection) can be simply viewed as $p_{ij} = 1$.

Finally, our formulation can be easily extended to Placement-only problem. It is defined to maximize β with respect to the decision variable u_{ij} only, while flowsets are routed along their original routes:

$$\texttt{Maximize } \beta \tag{2.26}$$

$$\beta = \sum_{y \in \theta} \sum_{(i,j) \in E} \mathcal{I}_y p_{ij} u_{ij} \Gamma^x_{y \in \Upsilon_x} \tag{2.27}$$

$$u_{ij} \in \{0, 1\}, \forall (i, j) \in E \tag{2.28}$$

2.4 MMPR Solutions

In this section, we first describe the optimal MMPR solution by solving the associated MILP problem in Section 2.4.1. Since the time-complexity of MILP is generally NP-hard, we propose several heuristic solutions to approximate the optimal performance: “K-Best", “Successive Selection", “Greedy" and “Quasi-Greedy". It is easy to see that MMPR becomes a LP (Linear Programming) problem if the monitor placement strategy is given (i.e., with fixed u_{ij}). Therefore, all of our heuristic solutions tend to decide the monitor locations first. They all start from an initial configuration in which all M monitors are fully deployed. We refer to this initial configuration as the “All-On" stage.

In particular, we first propose K-Best (Section 2.4.2), the most lightweight algorithm among our heuristic methods. It directly disables $M - K$ monitors according to their performance in the All-On case, based on some ranking metrics (e.g., traffic amount, topology, link capacity, etc). We then propose several increasingly complex algorithms, “Successive Selection", “Greedy", and “Quasi-Greedy", that iteratively

select monitors to disable, based on the planned monitor placement strategy decided from the previous iteration. This process is repeated until only K monitors are left. The Successive Selection algorithm (Section 2.4.3) uses the same heuristic metrics as K-Best to successively disable monitors at each iteration. The Greedy and Quasi-Greedy (Section 2.4.4) algorithms are the most complex since they select monitors to disable in each iteration by testing them.

All the proposed heuristics seek the least important monitors (in accordance to some metric) to disable and then maximize the measurement gain β. They all start from the All-On stage and gradually exclude monitors until K are left. Our approach is complementary to previous work on monitor placement [20, 72] that starts with zero monitors and gradually add new monitors until there are K of them. The reason for our design is the following: whenever monitors are chosen, the best routing for the flowsets needs to be re-calculated, which may change substantially after new monitors are introduced. Instead of testing possible placement and flowset routing, it is more straightforward to disable unimportant monitors from a stage with more enabled monitors. We therefore propose algorithms that start from the All-On stage.

2.4.1 Optimal Solution

The optimal solution searches for the best γ_{ij}^{y} and u_{ij} assignments for the MMPR problem. The MMPR formulation is an MILP problem since u_{ij} is a binary decision variable and γ_{ij}^{y} is a continuous decision variable. There is a variety of optimization tools that we can leverage. In particular, the optimal solution can be found using an MILP solver (e.g., CPLEX [4]). We refer to this solution as “Optimal". For small to medium size networks, the optimal MMPR solution can be readily found. However, given that MILP problems are in general NP-hard, the solvers are not fast enough for large networks.

2.4.2 K-Best Algorithm

The K-best algorithm disables $M - K$ monitors in a single step, based on their performance in the All-On stage. It starts from the All-On configuration and calculates

the maximum achievable β and optimal traffic assignment γ_{ij}^{y}. It then ranks all monitors in ascending order using one of the following metrics and directly disables the top $M - K$ monitors:

- *Least-utility ($\sum_y p_{ij}\gamma_{ij}^{y}I_y$).* We disable the monitors with the least measurement utilities. Since measurement utility is the same as our optimization objective, we expect this metric will achieve the best β.
- *Least-traffic ($\sum_y \gamma_{ij}^{y}\phi_y$).* The intuition behind this metric is that the monitors with the least amount of traffic passing through them are also expected to have the least contribution to the overall measurement utility.
- *Least-importance ($\sum_y \gamma_{ij}^{y}I_y$).* This metric only considers the flowset importance, regardless of the sampling rate. It treats all flowset with the same traffic demand and all monitors with the same sampling rate.
- *Least-rate (p_{ij}).* We disable monitors with the least sampling rates since they are the least capable.
- *Least-neighbor ($\sum_{k:(k,i)\in E} 1 + \sum_{k:(j,k)\in E} 1$).* From a topology perspective, the monitors that are the least connected are also likely to provide the least amount of freedom to MMPR for routing optimization.

The K-Best algorithm greatly saves computation time since only two LP problems are involved. The first LP decides the γ_{ij}^{y} for the All-On stage. Ranked in ascending order using one of the above metrics, the top $M - K$ monitors are disabled. Then, with these $M - K$ monitors turned off, a second LP is solved to maximize β using MeasuRouting [63]. However, since K-Best ranks the importance of each monitor based on metrics evaluated from the initial All-On stage, the measurement gain is predicted to diverge from the optimal.

2.4.3 Successive Selection Algorithm

The Successive Selection algorithm also starts from the initial All-On configuration with all M monitors and iteratively chooses D monitors to disable. Here, we use the same five metrics introduced in Section 2.4.2. The selection of which D monitors to disable is based on the ranking of remaining monitors $\hat{M}$ using one of the five metrics.

Algorithm 1 Successive Selection Algorithm

1: **while** More than K monitors are left **do**
2: Maximize β by using all remaining monitors
3: find the corresponding γ_{ij}^{y}
4: **for** Each remaining monitor $(i, j) \in \hat{M}$ **do**
5: Calculate its performance metric for one of the five principles with γ_{ij}^{y}
6: **end for**
7: Disable D monitors with least performance-metric
8: **end while**

In particular, it disables D monitors based on their ranking calculated from the previous iteration (Line 7). This means we use the information from the previous iteration (i.e., planned routes γ_{ij}^{y}, etc.) to calculate the metric for each monitor in the current iteration (Line 5).

Note that if the metric used is either the “least-rate" or the “least-neighbor" metric, both Successive Selection and K-Best will have the same selection of monitors and measurement gain since the metrics do not involve γ_{ij}^{y}.

2.4.4 Greedy Algorithm

Similar to Successive Selection, the Greedy algorithm also disables D monitors in each iteration, until K monitors are left. However, it is more complicated since it tests all remaining monitors $\hat{M}$ in each iteration. In order to test a monitor, it re-computes the maximized β after turning it off (Line 2-7), which essentially involves using MeasuRouting [63] (Line 4). Based on the testing of every remaining monitor, it disables D of them that have least impact on β (Line 8).

Since the Greedy algorithm exhaustively tests individual monitors at each iteration, its performance is hypothesized to be close to the optimal solution. It is still suboptimal since it tests individual monitors instead of every possible combination. However, the algorithm remains computationally costly, since it tests $O(\hat{M})$ monitors with $O(\hat{M})$ LP problems in each iteration. For a moderate sized topology, an MILP solver can sometimes work faster than this greedy approach. To reduce the computation

Algorithm 2 Greedy Algorithm

1: **while** More than K monitors are left **do**
2: **for** Each remaining monitor $(i, j) \in \hat{M}$ **do**
3: Disable the monitor
4: Maximize β based on remaining monitors
5: Store β
6: Enable the monitor
7: **end for**
8: Find D monitors with largest $\beta \in \hat{M}$ when they are disabled
9: $\hat{M} \leftarrow \hat{M}/\{(i, j) \in D\}$
10: **end while**

time, we propose a less heavy-weighted algorithm called “Quasi-Greedy", which is a derivation of the Greedy algorithm. In Quasi-Greedy, instead of testing every remaining monitor, it only tests λ fraction candidates, where $0 < \lambda < 1$. We use C to denote candidate sets.

The candidates C are chosen based on the least-utility metric (Line 4), where utility is defined as $\sum_y p_{ij}\gamma_{ij}^y I_y$. It benchmarks how much utility a monitor measures (Line 3). In each iteration, the Quasi-Greedy algorithm re-computes all the corresponding β by turning off one-by-one the remaining monitors in C to find the least important D monitor to disable (Line 5-11). It then disables these chosen D monitors from the remaining monitor set, $\hat{M}$ (Line 12). Besides least-utility, candidates can also be identified by using other heuristic metrics defined for the K-Best algorithm (Line 3).

2.4.5 Algorithm Examples

Suppose we have $M = 32$, $K = 24$ and $D = 4$. The K-Best algorithm directly disables $8 = 32 - 24$ monitors in a single step. On the other hand, the Successive Selection algorithm involves two iterations. In each iteration, $D = 4$ monitors are selected for exclusion based on their performance in the previous iteration, in accordance to one of the metrics defined in Section 2.4.2. The Greedy (Quasi-Greedy) algorithm also disables $D = 4$ monitors in each iteration. However, it selects the least

Algorithm 3 Quasi-Greedy Algorithm (λ)

while More than K monitors are left **do**
  Maximize β by using all remaining monitors
  Calculate measurement utility of each monitor $(i,j) \in \hat{M}$
  Choose C=λ fraction remaining monitor $(i,j) \in \hat{M}$ as candidates
  for Each candidate monitor $\in C$ **do**
    Disable the monitor
    Maximize β based on remaining monitors
    Store β
    Enable the monitor
  end for
  Find D monitors $\in C$ with largest β when they are disabled
  $\hat{M} \leftarrow \hat{M}/\{(i,j) \in D\}$
end while

important 4 monitors based on testing every (candidate) monitor one-by-one and solving the corresponding LP problem in each iteration, which is very time consuming. For the Greedy algorithm, it involves solving 32 and 28 LP problems in the first and second iteration, respectively. For the Quasi-Greedy algorithm, it involves solving $32 \times \lambda$ and $28 \times \lambda$ LP problems in the first and second iteration, respectively.

2.5 Evaluation

In this section, we evaluate the performance of MMPR. The experiment settings are described in Section 2.5.1. Section 2.5.2 presents the traces and the metrics of performance/cost to benchmark MMPR solutions. Section 2.5.3 discusses our evaluation results in detail.

2.5.1 Experiment Settings

We first define the settings for individual flows. We denote the set of flows as $\mathcal{F}$, the traffic demand of flow f as b_f, and the importance of sampling it as i_f. We use $\upsilon_{y \in \theta}$ to

represent the set of flows that belong to the flowset y. For our evaluation, we specify the measurement utility function of each flowset to be the following: $\mathcal{I}_{y\in\theta} = \sum_{f\in\upsilon_y} i_f b_f$.

The importance of a flow f, i_f, can be viewed as points we earn if a byte of it is sampled. The optimization objective of MMPR is to maximize β. It is easy to see that β can be expressed in another way: $\sum_{f\in F} i_f b_f T_{\upsilon^{-1}(f)}$, which is exactly the total number of points earned by MMPR. Here $\upsilon^{-1}(f)$ denotes the flowset to which flow f belongs.

Most IP networks use link-state protocols such as OSPF [60] and IS-IS [43] for intra-domain routing. In such networks, every link is assigned a cost and traffic between any two nodes is routed along minimum cost paths. In this paper, we use the popular local search meta-heuristic in [34] to optimize link weights with respect to our aggregate traffic demands. The optimized link weights are then used to derive our original routing $\{\Gamma\}^{x\in\Theta}_{(i,j)\in E}$. To avoid randomness in [34], we conduct experiments for the same setting five times, and only show the average results.

We have the greatest degree of freedom if each flow is assigned to a unique flowset. However, this is not scalable from both a computation and an implementation perspective. Therefore, we have q flowsets per OD pair. We also have $L \geq q$ flows for each OD pair. Each of the L flows in $\mathcal{F}$ belonging to a particular OD pair is assigned to one of the flowsets. There can be multiple ways of making such an assignment. In this paper, we randomly assign an equal number of flows to each of the q flowsets.

Table 2.1 lists the values for the MMPR parameters used for all the experiments in Section 2.5.3. We generate sampling rates for each link using uniform distribution between 0 and 0.1. For one realization of link sampling rate and traffic demand, we repeat the experiments 10 times with different flow importance i_f generated from the Pareto distribution. We present the average measurement gain, unless specified otherwise. We use CPLEX [4] to find optimal solutions for the LP and MILP problems. For all the heuristic algorithms, we choose $D = 4$ and $D = 8$ in Abilene and AS6461/GEANT network, respectively. The algorithms with a larger D disable more monitors in each iteration. However, our evaluation results suggest that the performance is actually insensitive to the value of D, and the results are omitted here.

Table 2.2: Default Experimental Parameters

Parameter	Description	Value/Distribution
q	Flowsets per OD pair	10
ϵ	TE violation threshold	0.1
i_f	Flow importance	Pareto ($\lambda = 2$)

2.5.2 Traces and Performance Metrics

We use these three topologies in our experiments:

- *Abilene*: It is a public academic network in the U.S. with 11 nodes interconnected by 28 OC192 (10 Gbps) links. The traces used were from April 22-26, 2004 [1].
- *AS6461*: It is a RocketFuel [69] topology with 19 nodes and 68 links. To generate artificial traces, we first generate aggregate traffic demands for each OD pair using a Gravity Model [55]. The traffic demand of flow f, b_f, is then set equal to the traffic demand of its corresponding OD pair divided by L, where $L = 3000$.
- *GEANT*: It connects a variety of European research and education networks. Our experiments are based on the December 2004 snapshot [3], which consists of 23 nodes and 74 links ranging from 155 Mbps to 10 Gbps.

In our experiments, besides the measurement gain β and the TE metric in terms of MLU (maximum link utilization), we are also interested in the following four performance metrics:

- *Computation Time*: In our experiment, we only collect computation time for the LP or MILP solver. These parts usually take longer time than normal numerical computation, and are therefore the dominant part for our solutions. Meanwhile, the computation time for LP or MILP may vary for different solvers. We therefore do not mix them with other numerical computation.
- *F&T TE metric*: We use MLU as the TE metric in Equation (19). Besides MLU, we are also interested in the F&T metric [34], which is defined as weighted summation of link utilization of all the links. F&T characterizes the performance of entire network.

- *APLI(Average Path Length Inflation)*: It is defined as the ratio of $\sum_y \sum_{(i,j)\in E} \gamma^y_{(i,j)} \phi_y$ and $\sum_y \sum_{(i,j)\in E} \Gamma^x_{(i,j)} \phi_y$. APLI reflects how flows get detoured. We expect important flows to have large path inflation since they are re-routed towards monitors.
- *Monitor Selection Overlap* η: It is defined as a ratio. The numerator is the number of monitors that are both selected by the heuristic and optimal solution. The denominator is the total number of selected monitors. A key part of the MMPR problem is to select the best monitor locations. This η metric reflects how heuristics select monitors.

2.5.3 Evaluation Results

In this section, we first compare Optimal MMPR with two baseline cases in Section 2.5.3 and show that MMPR can have better measurement gain up to 1.17X and 2.6X when compared to Placement-only and MR-only, respectively, for Abilene, 1.71X and 6X for AS6461, and 1.14X and 6.6X for GEANT. Section 2.5.3 presents detailed performance comparison of our proposed heuristic algorithms belonging to three different categories: K-Best, Successive Selection and Quasi-Greedy. Section 2.5.3 shows that all our proposed heuristic algorithms in each category perform very close to the Optimal MMPR solution and can reduce the computation by a factor of 23X in Abilene, 246X in AS6461, and 233X in GEANT.

In all the figures below, we use “KB", “SS", and “QG" to denote K-Best, Successive Selection and Quasi-Greedy, respectively. For example, “KB/utility" means K-Best algorithm with the least-utility ranking metric. Optimal MMPR is denoted as “Optimal" for short. For all the figures on computation time, the unit is second.

Optimal Solution vs. Default Cases

We first compare the optimal solution of MMPR with MR-only and Placement-only, using the same experimental settings (Section 2.5.1). Placement-only was defined and formulated in Section 2.2. MR-only, on the other hand, first randomly selects K monitors, and then finds the optimal routing γ^y_{ij} using MeasuRouting [63]. As shown

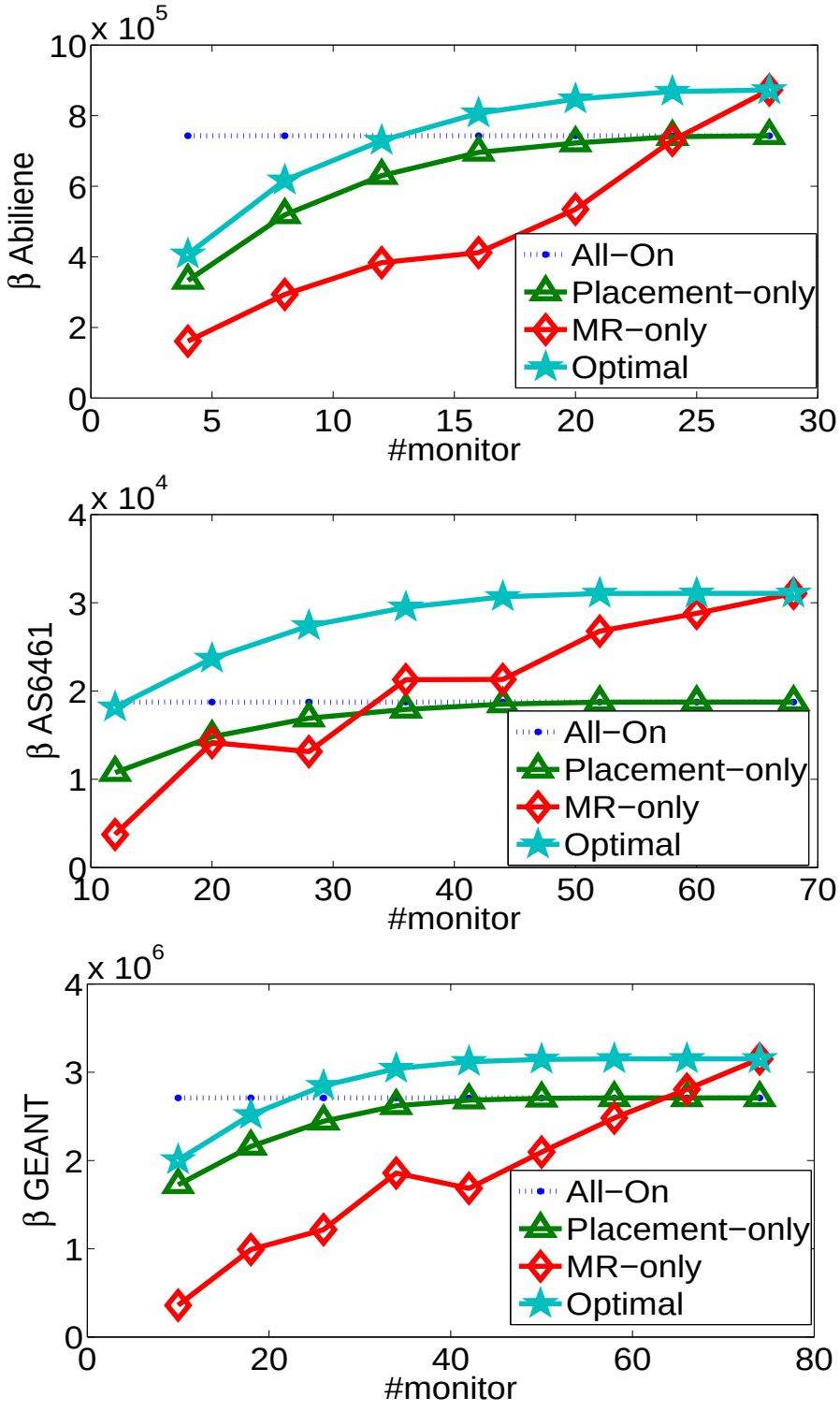


Figure 2.2: Compare Optimal MMPR with Default Cases for Abilene, AS6461 and GEANT

in Figure 2.2. We see that optimal MMPR can have better measurement gain up to 1.17X($\frac{87}{74}$) and 2.6X($\frac{4}{1.5}$) compared to Placement-only and MR-only, respectively, for Abilene, 1.71X($\frac{3}{1.75}$) and 6X($\frac{1.8}{0.3}$) for AS6461, and 1.14X($\frac{3.1}{2.7}$) and 6.6X($\frac{2}{0.3}$) for GEANT.

We also present the performance of another baseline case, “All-On", in which every monitor is on and flowsets are routed by the default routing $\Gamma^x_{(i,j)}$. As shown in Figure 2.2, the optimal β of MMPR is better than the “All-On" case, even with only a small fraction of monitors turned on. Without strategic routing, even deploying monitors everywhere does not guarantee a comparable performance gain compared with MMPR with a small number of monitors. As shown in these figures, MMPR can achieve the same measurement gain as the “All-On" case, but it can save 16(=28-12), 56(=68-12), and 54(=74-20) monitors in the case of Abilene, AS6461, and GEANT, respectively. Meanwhile, the computation time for optimal solution is fairly long (around 4 minutes) for AS6461, and increases to around 6 minutes for the GEANT network.

Sensitivity Analysis of Heuristic Algorithms

Due to the potentially long computation times required to solve for the optimal MMPR, we propose several heuristic algorithms to reduce the computation time complexity. They are categorized as “K-Best", “Successive Selection", “Greedy" and “Quasi-Greedy". We omit performance results for Greedy since it is computationally too costly.

We first compare K-Best algorithms (using different metrics) with the optimal solution in Figure 2.3. As expected, using the least-utility metric achieves the best β (very close to optimal) in all three topologies. It achieves 2.36X($\frac{2.6}{1.1}$) higher measurement gain compared to using the least-importance metric, but only increases 1.1X($\frac{1.22}{1.1}$) in computation time in AS6461.

As mentioned earlier, the computation time is only collected for the LP or MILP solver. Results in Figure 2.4 show that using different ranking metrics lead to very similar computation times. From the perspective of an LP solver, an unsuitable monitor placement means either more steps are needed to achieve the optimal β (which is more time consuming), or there is no way to achieve very large β (which means shorter solving time). If we also consider other numerical computations (i.e: computation of

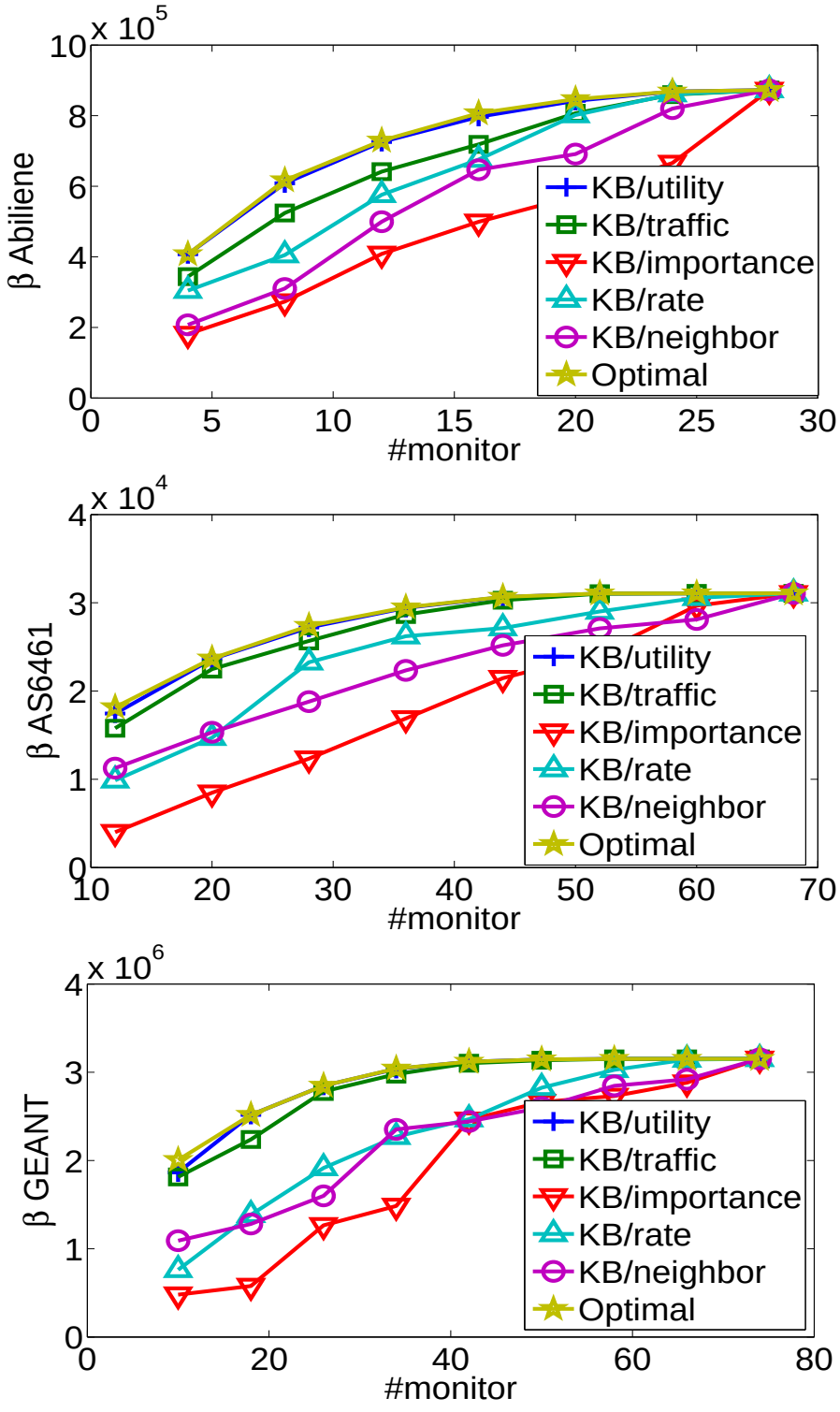


Figure 2.3: MMPR Performance for K-Best Algorithms in Abilene, AS6461 and GEANT

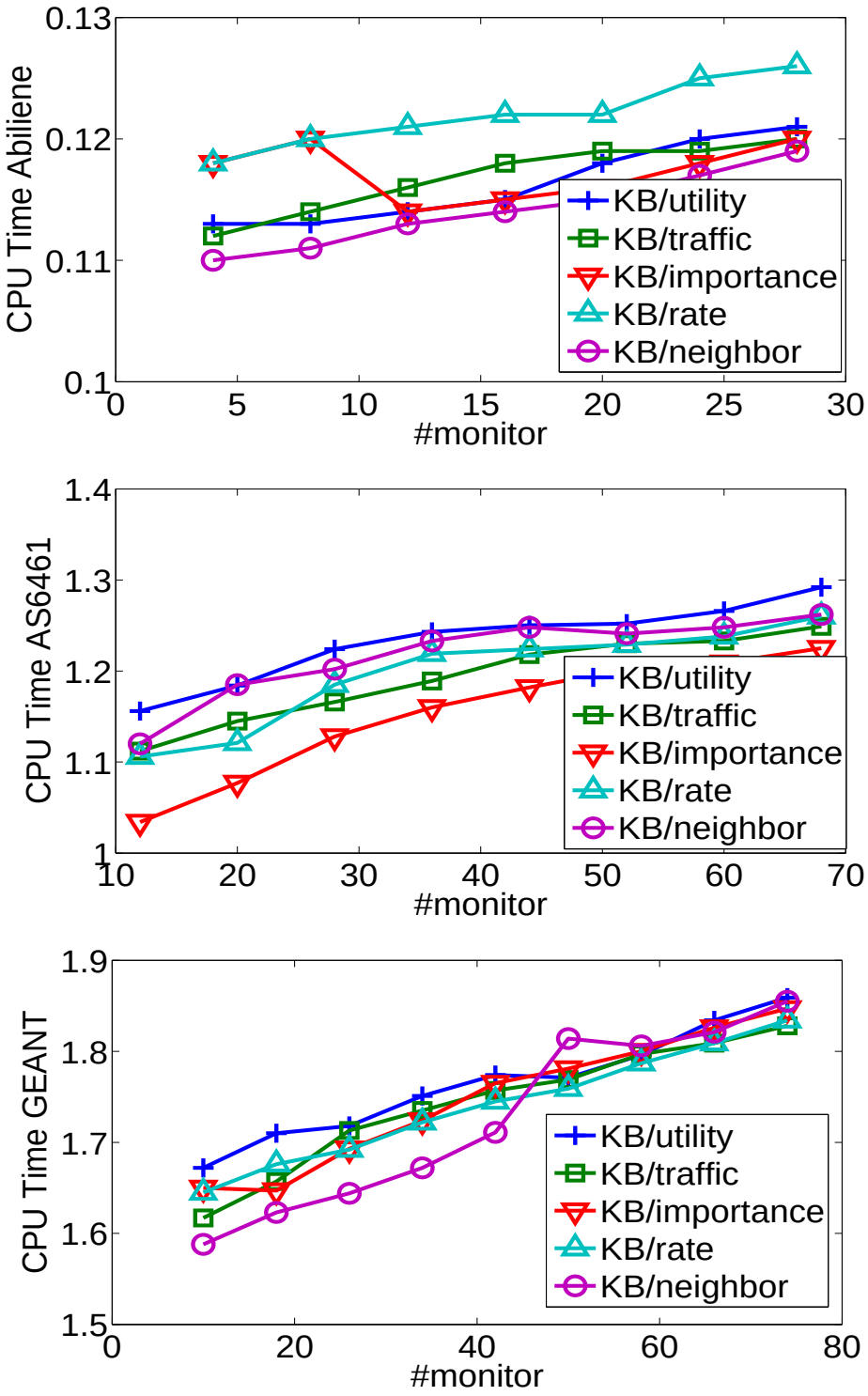


Figure 2.4: MMPR Computation time for K-Best Algorithms in Abilene, AS6461 and GEANT

each metric, ranking monitors based on metric values), “least-utility", “least-traffic", and “least-importance" definitely take longer, since the calculation of these three metrics involve all flows and monitors. In contrast, “least-rate" and “least-neighbor" only need topology information.

Figure 2.5 compares the MMPR performance of Successive Selection algorithms with different ranking metrics, and the same trend is observed in Abilene, AS6461, and GEANT networks. We omit “least-rate" and “least-neighbor" cases since they have similar measurement gain as in the corresponding K-Best case. Successive Selection with the least-utility metric also achieves the best performance. Similar to Figure 2.4, the three metrics share very close computation time (e.g., shown in Figure 2.6). It mostly depends on the number of iterations, which is linear with respect to the number of monitors in the Successive Selection algorithm.

Finally, we compare the Quasi-Greedy algorithm (with different λ values) against the optimal solution in Figure 2.7. Since Quasi-Greedy is still computationally intensive, we only present results for AS6461. Note that there are no obvious improvements on measurement gain for larger λ's. However, the computation time increases substantially with larger λ's. This implies that even with a smaller number of candidates, the Quasi-Greedy algorithm can perform very close to the optimal and saves computation time.

Comparing K-Best, SS, and QG

In this section, we compare all three heuristic algorithms with the optimal MMPR solution. Results from the previous section show that “least-utility" is the most effective metric for ranking the importance of monitors. We therefore adopt “least-utility” metric as a basis for comparing the K-Best, Successive Selection, and Quasi-Greedy methods.

For the Quasi-Greedy algorithm, we present results using $\lambda = 0.15$. It tests $0.15\bar{M}$ monitors in each iteration to choose D monitors to disable. Figure 2.8 and Figure 2.9 show the achieved measurement gain β and computation time for all the algorithms for all three topologies. In addition, we present F&T metric, APLI (Average Path Length Inflation), and η (Monitor Selection Overlap) in Figure 2.10. Only results

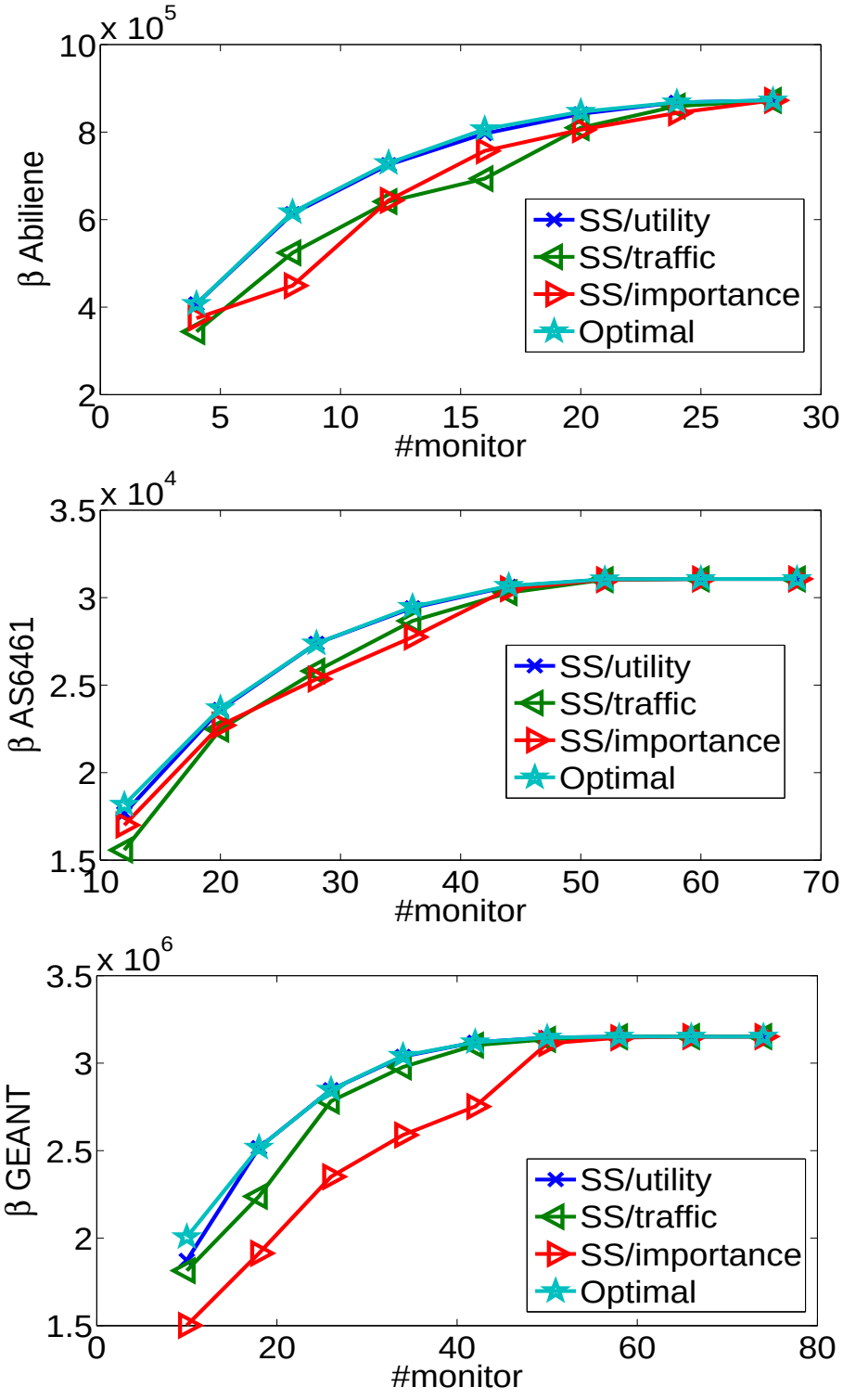


Figure 2.5: MMPR Performance for Successive Selection Algorithms in Abilene, AS6461 and GEANT

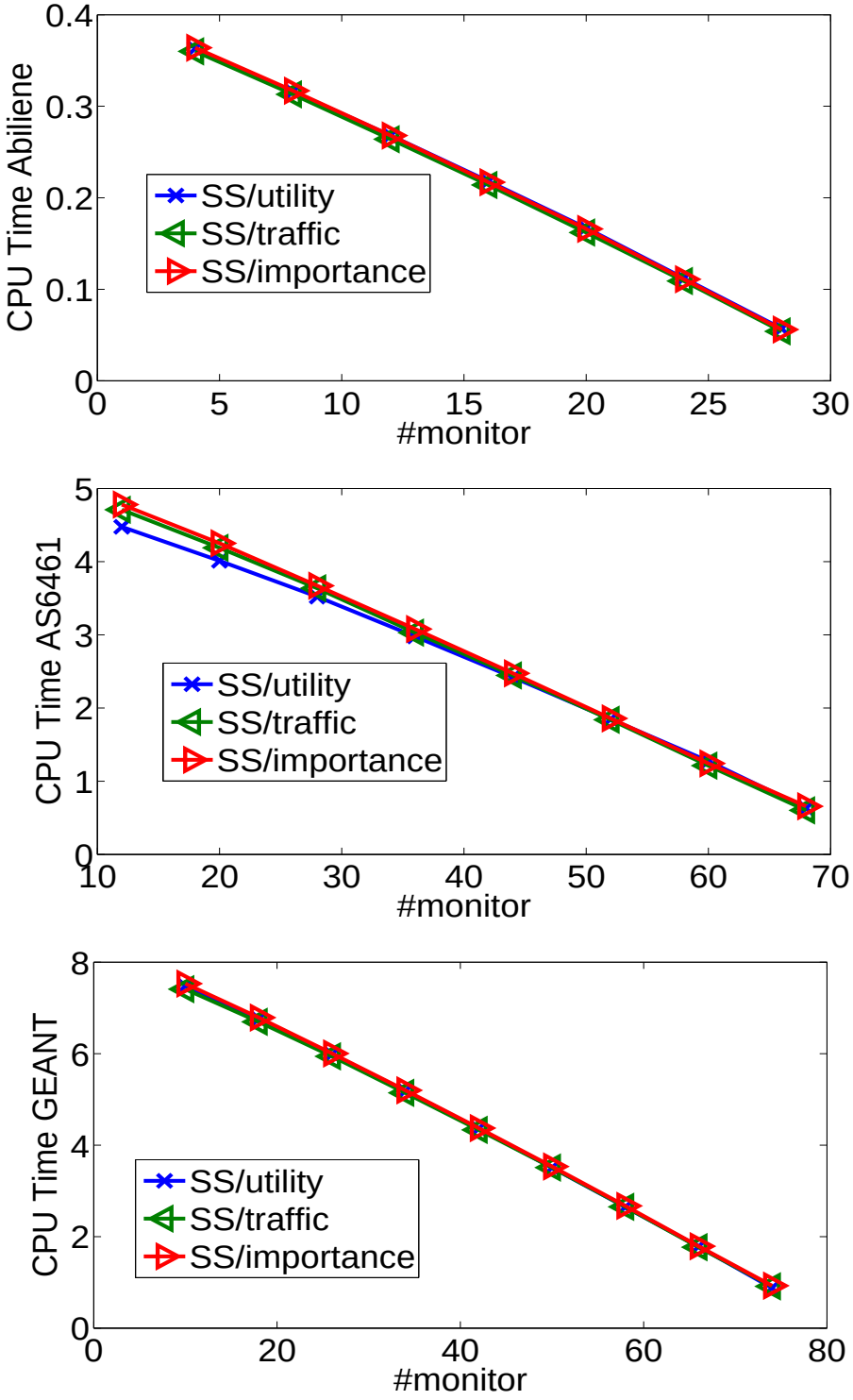


Figure 2.6: MMPR Computation time for Successive Selection Algorithms in Abilene, AS6461 and GEANT

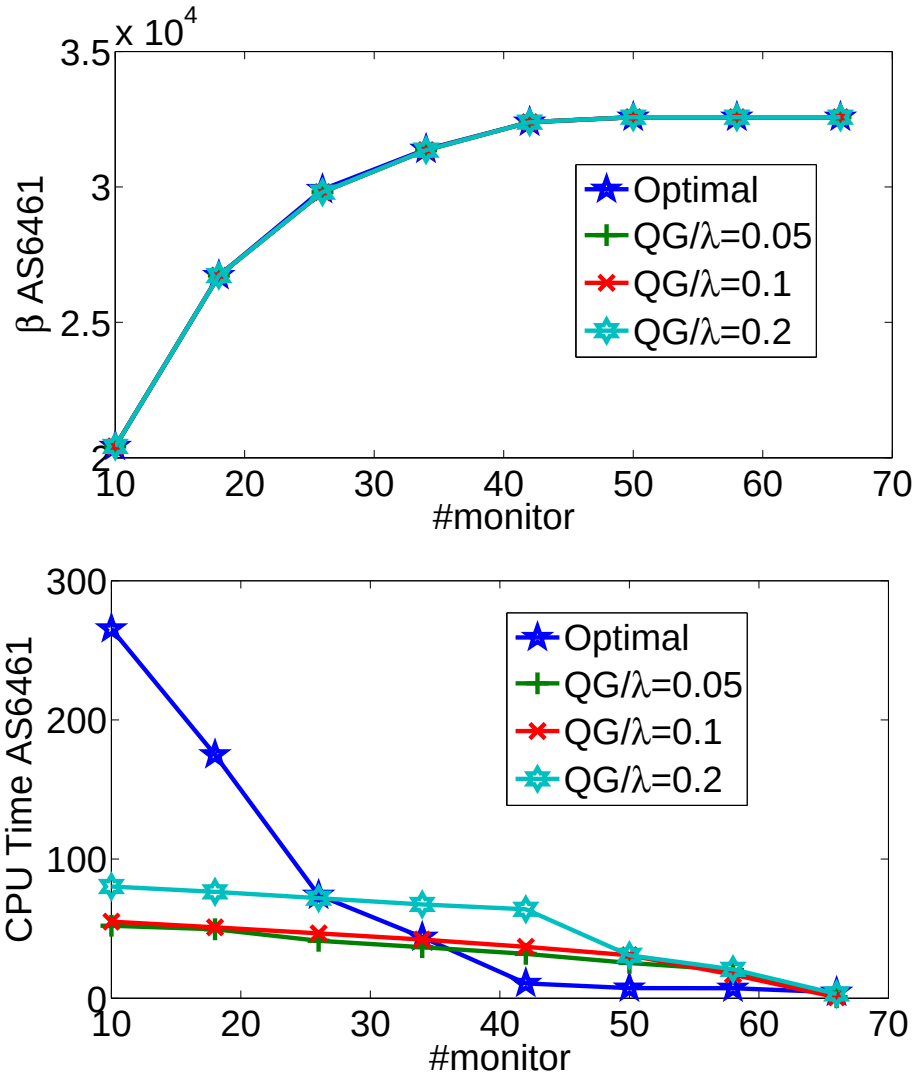


Figure 2.7: MMPR Performance for Quasi-Greedy Algorithm for AS6461

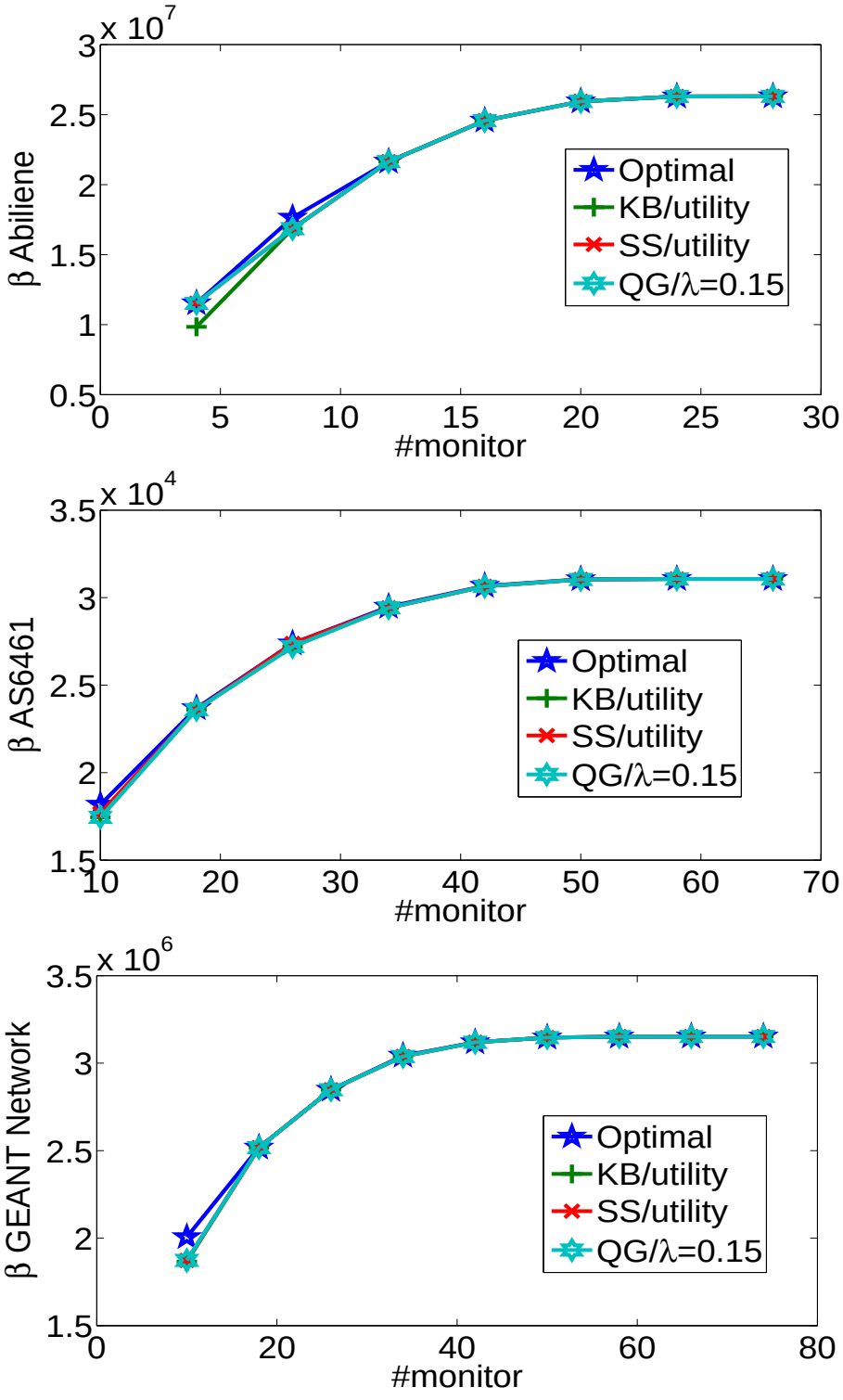


Figure 2.8: MMPR Performance for Heuristic Algorithms in Abilene,AS6461 and GEANT

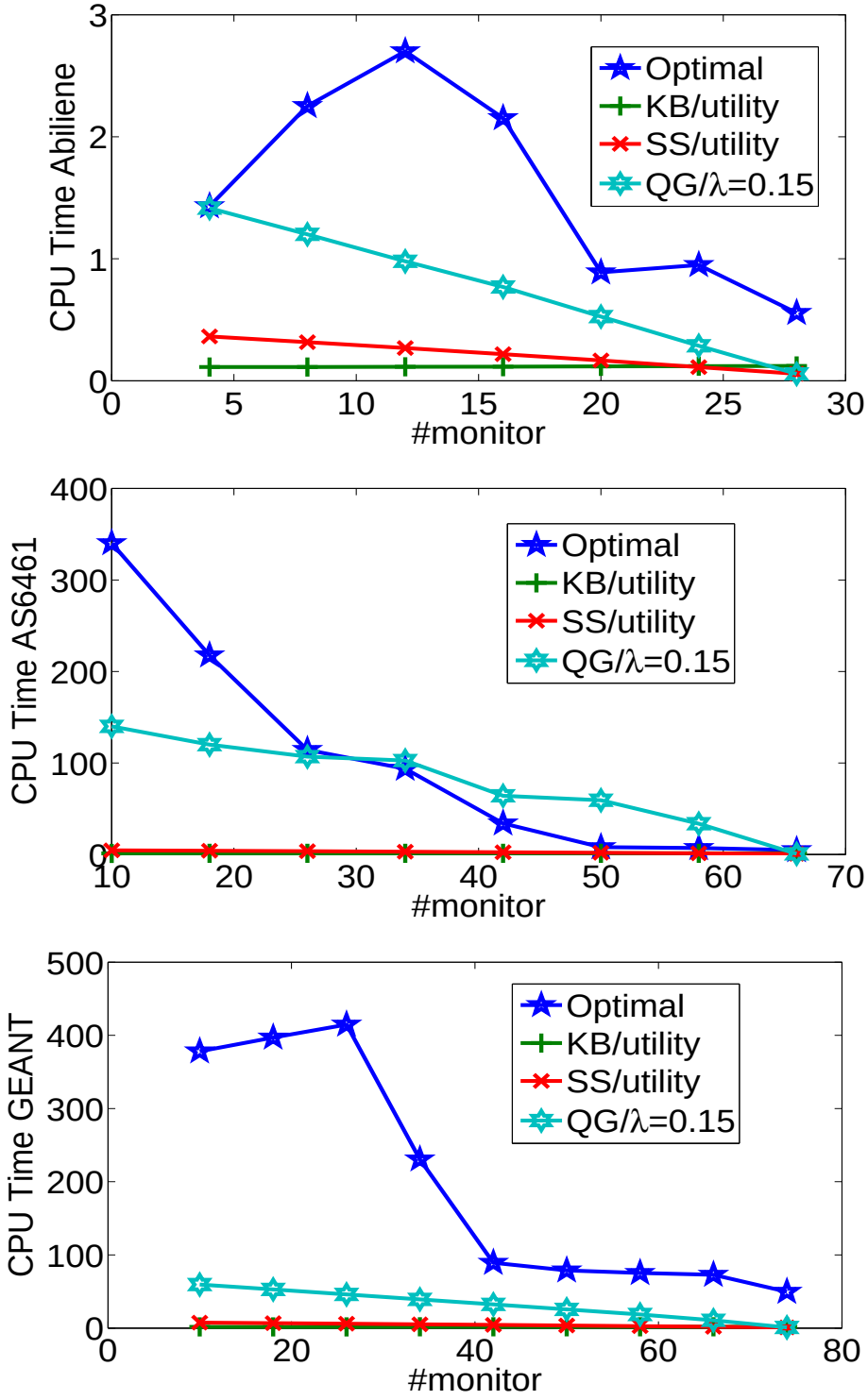


Figure 2.9: MMPR Computation time for Heuristic Algorithms in Abilene,AS6461 and GEANT

for AS6461 are shown, but the same trends are observed for the other two topologies. We make the following observations based on our results:

- The maximum β's are very close for all algorithms. Both K-Best and Successive Selection algorithms are practical for large networks; their computation times are much less than the optimal case. Their best metric is "least-utility". Although K-Best is slightly worse than Successive Selection for Abilene, their achievable β's are almost the same for a large network like GEANT.
- Quasi-Greedy approach is very costly. However, its measurement gains are not noticeably better than the other heuristics. Therefore, there is no need to iteratively test monitors one-by-one to decide which ones to disable. We can just simply disable monitors based on their performance metrics in the previous iteration.
- As shown in Figure 2.8, K-Best achieves almost the same measurement gain as MMPR optimal, but reduces computation times by a factor of 23X($\frac{2.7}{0.12}$), 246X($\frac{320}{1.3}$), and 233X($\frac{400}{1.71}$) for Abilene, AS6461, and GEANT, respectively, while Successive Selection reduces computation times by a factor of 10X($\frac{2.7}{0.26}$), 64X($\frac{320}{5}$), and 66X($\frac{400}{6}$) for Abilene, AS6461, and GEANT, respectively. Quasi-Greedy also saves computation times by a factor of 3X($\frac{300}{100}$) for AS6461. In practice, K-Best is the best choice since it greatly reduces computation time with measurement gains that are very close to the optimal.
- Values for F&T metric and APLI both increase with larger number of monitors. With more monitors, MMPR will put more weight on improving measurement gains, at a cost to the traffic engineering and packet forwarding performance. For example, because the same threshold $\epsilon = 0.1$ is used to bound TE violation in Equation (2.19), all algorithms finally achieve the same MLU in every case (graphs are omitted here). However, both F&T metric and APLI increase with more monitors. For example, the 20% increase in APLI implies longer end-to-end forwarding delay, which may be acceptable for non-real-time traffic. To meet more stringent QoS requirements, they can be introduced as constraints in the MMPR formulation.
- The optimal solution does not necessarily achieve the best F&T or APLI results, since the optimal solution only optimizes for measurement gains with bounded violation of MLU. Some of the heuristics work better in preserving the overall

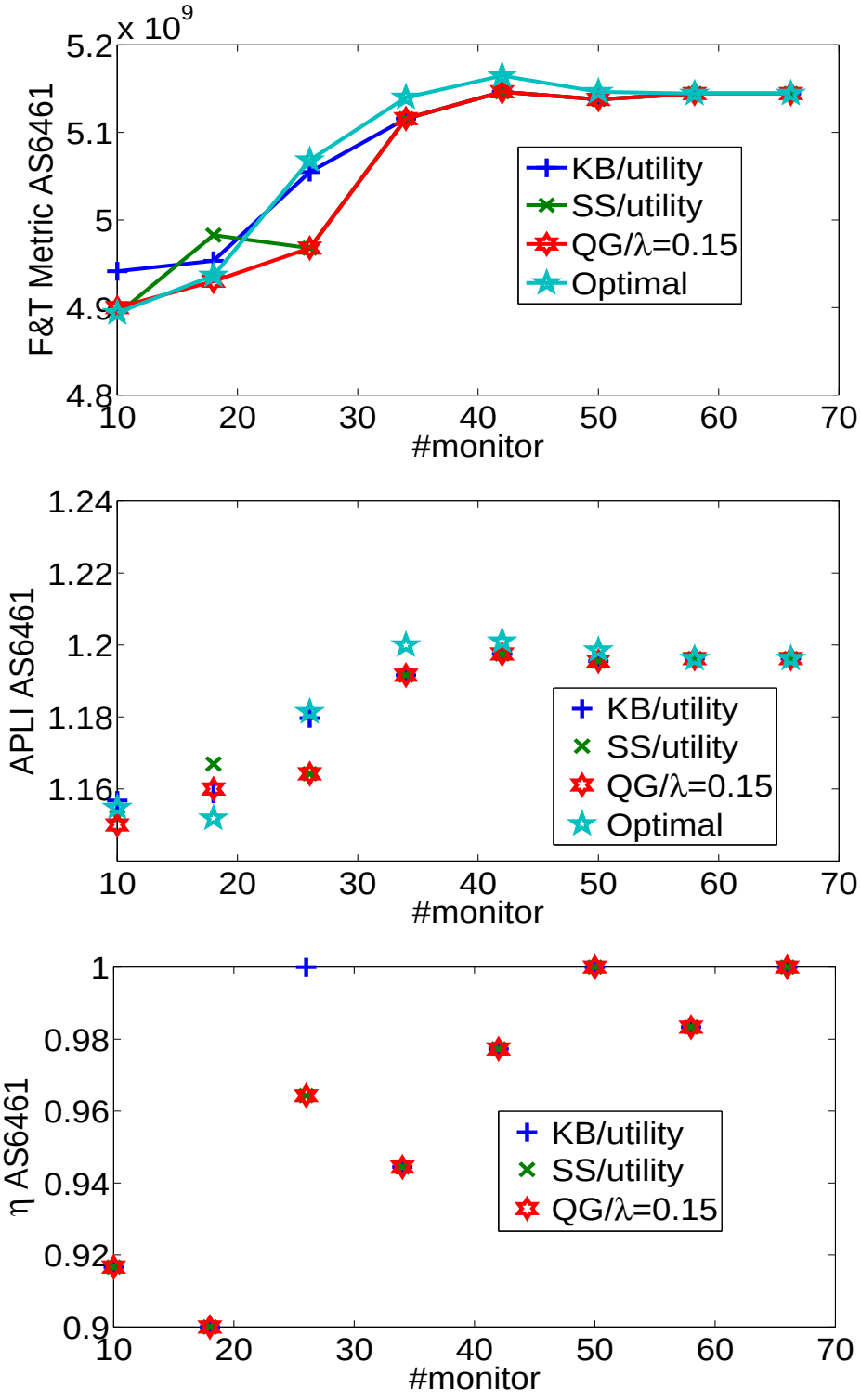


Figure 2.10: Compare Other Performance Metrics for AS6461

network performance.

- η (Monitor Selection Overlap) shown in Figure 2.10 for AS6461 provides insights into why the performance of different algorithms are so close to the optimal. All the heuristic algorithms select almost the same set of monitor locations (e.g., 92%-100%) as the optimal solution, with the ratio approaching one as the number of available monitors increases. The same trend is observed for Abilene and GEANT topologies (results not shown here).

2.6 Related Work

Previous work mostly studied traffic measurement on a single monitor. They either infer traffic characteristics from sampled data [21, 22, 29, 37, 54] or use measurement schemes other than sampling for special traffic sub-populations [30, 50, 52, 53, 62, 75].

Recently, researchers have begun investigating network-wide traffic measurement problems. Existing approaches [15, 20, 72] generally define and solve some monitor placement problem for fixed traffic characteristics and monitoring objectives. [20] defines utility functions for the sampled traffic. The problem is to maximize the overall utility with bounded measurement operation/deployment cost. It models variations of this problem, proves their complexities, and proposes heuristic algorithms. [15] improves upon [20] by performing a more rigorous analysis to indicate the convergence of any heuristic solution. Most recently, [65] proposes Successive c-Optimal design to optimize the deployment and sampling rate of large IP networks. However, their measurement goal is traffic matrix estimation. In contrast, MMPR is not restricted to any special measurement goal. None of them are suitable for changing traffic conditions or monitoring objectives.

Our work builds upon the recently proposed MeasuRouting paradigm [63], which proposes to assist traffic monitoring by intelligently routing traffic sub-populations over the corresponding monitors. It assumes fixed and random monitor placement, and routes flowsets based on their different measurement importance. It maximizes the overall measurement gain β under the constraint that σ is preserved

at decent levels. With the freedom of intelligent routing, flows can better utilize the existing monitor infrastructure. Our work extends this framework by carefully choosing monitor locations. Our formulation also builds upon CSamp [66] like methods, to ensure non-overlapping measurement across monitors. [66] sets distinct hash filters on each monitor such that they capture different traffic sub-populations.

2.7 Discussion and Conclusions

In this chapter, we presented MMPR, a theoretical framework that jointly optimizes monitor placement and dynamic routing strategy to achieve maximum measurement utility, with limited monitoring resources. We formulated optimal MMPR as an MILP problem and proposed four heuristic algorithms to reduce the computation complexity: “K-Best", “Successive Selection", “Greedy" and “Quasi-Greedy". We performed detailed comparative study of these algorithms on three topologies, using both real traces and synthetic data. Our results suggest that the simplest algorithm, “K-Best", is actually the best choice in practice. It achieves measurement gains that are quite close to the optimal solutions, but it reduces the computation time by a factor of 246X in the best case in our experiments.

The theoretical study of MMPR framework can be extended by introducing other constraints or variations. For instance, as discussed earlier, measurement deployment/operation cost can be formulated in more concrete forms. Meanwhile, how to decide the proper flow utility function and measurement objective function remain open problems across different measurement applications. Furthermore, it would be interesting to treat sampling rate as another degree of freedom [15, 20], to let monitors dynamically adjust their monitoring capability. All these issues will be explored in future work.

MMPR, as well as MeasuRouting, require the prior knowledge of traffic importance in order to route flowsets differently. Such information need not be accurate in practice. Real measurements usually conduct hypothesis test procedures [71, 75]. They first obtain some global knowledge of the traffic, and zoom into the suspected traffic sub-population for more detailed analysis. Consider measuring flow size

distribution for small/medium flows as our target, MMPR/MeasuRouting can depend on external modules to first estimate large flow identities. MMPR/MeasuRouting then directs the large flows away from measurement boxes, which are devised with many small-sized counters such that small/medium flows can be better maintained. In this example, there is no need to accurately measure flow sizes in the first step.

There are also many implementation issues of MMPR that need to be addressed. One important issue is to determine which exact routing protocols are used. A routing protocol that strictly routes traffic between an OD pair along only the shortest paths may provide less opportunities to 're-route' important flowsets through monitors. A centralized routing controller, e.g. [59], is able to detour flows away from the shortest path. Meanwhile, MMPR requires that the traffic be dynamically routed/rerouted. Such dynamic forwarding mechanism can be implemented using programmable routers [13, 57, 59]. Besides this, two other dynamics issues are also important: how to estimate flow importance dynamically and how configure routing table entries dynamically. Recent work in [41] summarizes these challenges for MeasuRouting and proposes corresponding solutions for one measurement application: global iceberg detection and capture. The solutions are also applicable to MMPR, which builds upon MeasuRouting. MMPR extends MeasuRouting by introducing the opportunity to turn on and off monitors. In reality, operators should avoid frequently switching monitor status. We plan to implement MMPR in OpenFlow [59] or other programmable routing platforms in future work.

Chapter 2, in full, is a reprint of the material as it appears in IEEE Transactions on Network and Service Management (TNSM) 2011, Guanyao Huang, Chia-Wei Chang, Chen-Nee Chuah and Bill Lin, "Measurement-aware Monitor Placement and Routing: A Joint Optimization Approach for Network-Wide Measurements". The dissertation author was the primary investigator and second author of the paper.

Chapter 3

LEISURE: A Framework for Load-Balanced Network-Wide Traffic Measurement

3.1 Introduction

Accurate traffic measurement is essential to a variety of network management tasks, including traffic engineering (TE), capacity planning, accounting, anomaly detection, and security forensics. Many existing studies focus on improving traffic measurement techniques at a single monitor, including adaptive sampling [61], data streaming [47], and heavy-hitter detection mechanisms [30]. These solutions typically examine packet headers to determine if any statistics need to be collected. While these aggregate traffic volume statistics are sufficient for TE purposes, there is an increasing need for fine-grained flow level measurements to perform accurate traffic classifications for security purposes. For example, deep packet inspection (DPI) allows post-mortem analysis of network events and helps understand the payload properties of transiting Internet traffic. Another solution, Network DVR [17], performs selective flow-based trace collection by matching packets against application-specific signatures.

However, doing fine-grained flow level measurements (e.g., analyzing payload) is often an expensive process that requires dedicated hardware (e.g., TCAMs [74]),

specialized algorithms, (e.g., Bloom Filters [25]), or vast storage capacity. Given the fast-changing Internet traffic landscape and large traffic volume, a single monitor is not capable of accomplishing the measurement tasks from all applications of interest due to its resource constraint. This calls for coordinated measurement between multiple distributed monitors. Moreover, network-wide traffic measurement at multiple monitors is also key to uncovering global network behavior since traffic measured at a single monitor only provides partial views and may not be sufficient or accurate. For example, a *global iceberg* [40] may have high aggregate volume across many different monitors, but may not be detectable at any single monitor. Discovering this type of event is important for a number of applications (e.g. detecting DDoS attacks, discovering worms, as well as ensuring SLA compliance).

To perform effective network-wide traffic measurement across multiple distributed monitors, a centralized framework that coordinates measurement responsibilities across different monitors is needed. In today's network, deployed monitors measure traffic completely independently to each other, leading to redundant flow measurements and inefficient use of routers' measurement resources. Sekar et al. [66] proposed CSAMP (Coordinated Sampling), a centralized hash-based packet selection system as a router-level primitive, to allow distributed monitors to measure disjoint sets of traffic without requiring explicit communications, thus eliminating redundant and possibly ambiguous measurements across the network. CSAMP uses an optimization framework to specify the set of flows that each monitor is required to record by considering a hybrid measurement objective that maximizes the total flow-coverage subject to ensuring that the optimal minimum fractional coverage of the task can be achieved. However, both traffic characteristics and measurement tasks can dynamically change over time, coupled with ever-increasing link rates (high traffic volume) and out of consideration to distribute multiple measurement tasks jointly, rendering previously-placed monitors easily overwhelmed if the measurement tasks are not judiciously load-balanced across them, thus leading to entire coordinated measurements failure and wastage of routers' measurement resources. In addition, existing frameworks (e.g., CSAMP) are agnostic to differentiation in the importance of traffic sub-populations or the cost of individual measurement tasks.

We present a new centralized optimization framework called *LEISURE* (**L**oad-**E**qual**I**zed mea**SURE**ment) to address the network measurement load-balancing problem on various realistic scenarios while ensuring that the maximum traffic measurement utility of the network is achieved. In contrast to CSAMP, LEISURE distributes traffic measurement tasks evenly across coordinated monitors subject to ensuring that the required fractional coverage of those tasks (e.g., the maximum traffic measurement gain from MeasuRouting [63], MMPR [39]) can be achieved. It takes a) routing matrix, b) the topology and monitoring infrastructure deployment and c) measurement requirements of tasks as inputs, and decides which available monitors should participate in each specific measurement task and how much they need to measure to optimize the load-balancing objectives. Ideally the *load-balancing objective* is to have identical workload for all monitors where workload denotes the normalized traffic amount that each monitor measures. In this work, the *load-balancing objective* is mainly defined as two terms: 1) minimizing the variance of workloads across all monitors or 2) minimizing the maximum workload among them. We summarize our contributions as follows:

- We present LEISURE and formulate the optimization problems for network-wide traffic measurements by considering different load-balancing objectives without compromising on the overall maximum traffic measurement gain of the network. The optimal solutions are translated into the disjoint sets of required-measured flows that each monitor is assigned to measure. We also propose simple heuristic solutions to compare with and extend LEISURE to incorporate practical scenarios (constraints), i.e., (a) with limited measuring resources at monitors, (b) with limited number of deployed monitors, (c) with multiple routing paths (e.g., ECMP) for each origin-destination (OD)-pair traffic.

- As proof of concept, we perform detailed simulation studies based on Abilene [16] and GEANT [73] network topologies and traces. Our results show that the significant load-balancing improvement (e.g., 4.75X smaller maximum workload and 70X smaller variance in workloads) is achieved by using LEISURE to optimally distribute the measurement tasks across all coordinated monitors when compared with the naive uniform assignments.

- We also present detailed performance comparison of our proposed heuristic algorithms belonging to two categories: LB-Greedy and LB-Successive Selection in flexible monitor deployment scenario. We show that our proposed heuristic solutions can achieve load-balancing performance that are quite close to the optimal solutions, while reducing the computation times by a factor up to 22.5X in Abilene and 800X in GEANT.

- We extend LEISURE and simulation studies to perform optimizations and sensitivity analysis with respect to multiple measurement tasks that exhibit different importance and incur different costs. We show that LEISURE is flexible enough to assign the correct set of measurement tasks for coordinated monitors to optimize measurement utility given limited measuring resources.

This chapter is structured as follows. Section 3.2 outlines related work. Section 3.3 motivates our load-balancing problem by showing how measurement tasks can be distributed to several coordinated monitors using diversity of intuitions. We present detailed optimization formulations, solutions and implementations in Section 3.4, followed by the discussion of extensions in Section 3.5. Section 3.6 describes our simulation setup and evaluation results, and Section 3.7 concludes this chapter.

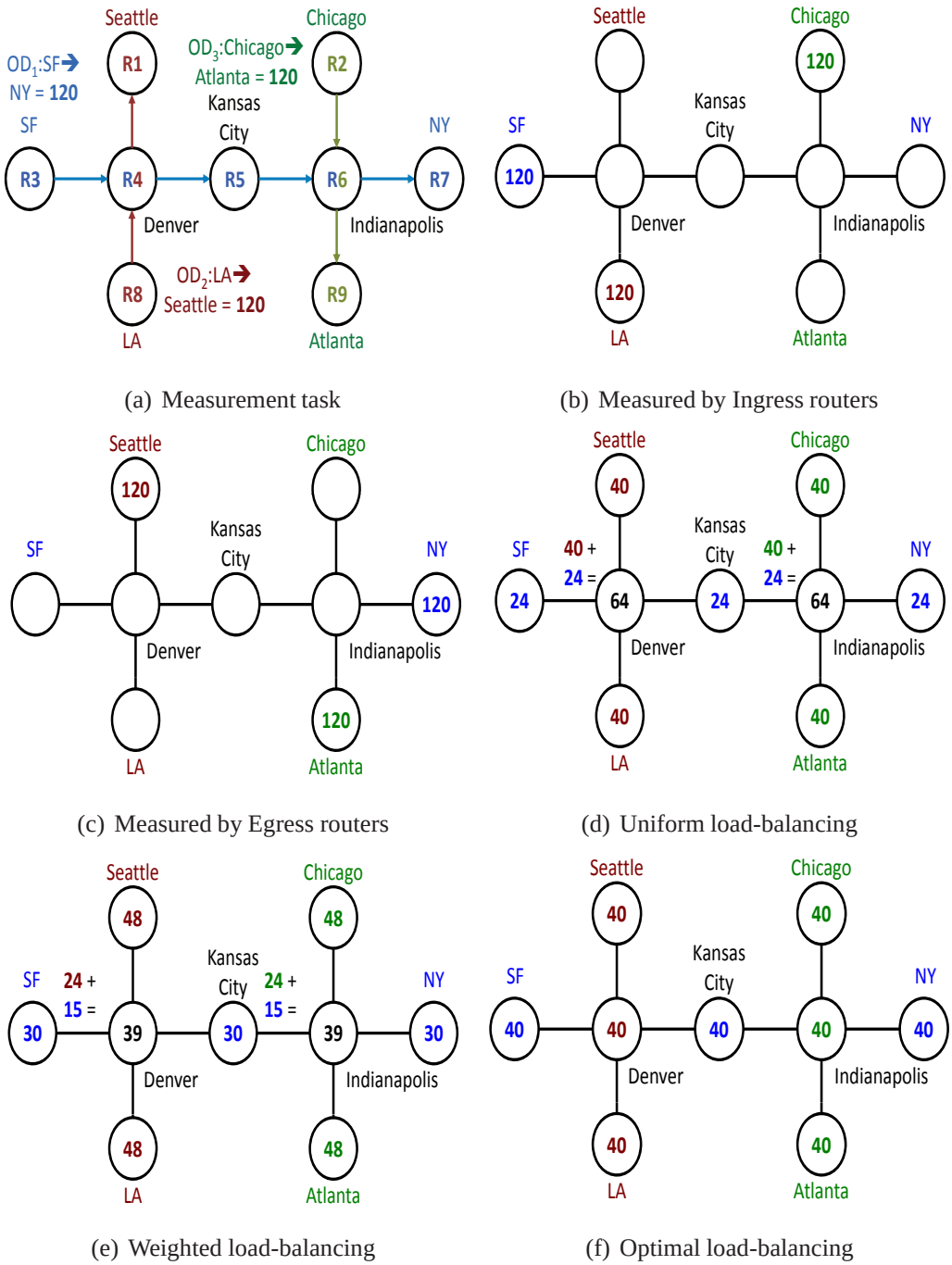
3.2 Related Work

Traffic measurement might involve single point or multiple monitors. Earlier work on traffic measurement has focused on improving single-point measurement techniques, such as sampling approaches [22, 37], estimation of heavy-hitters [30], and methods to channel monitoring resources on traffic sub-populations [62, 75]. Recently, researchers are interested in investigating network-wide traffic measurement problems. In particular, they have demonstrated the benefits of a network-wide approach for traffic engineering [76] and network diagnosis [48].

Network-wide traffic measurement presents more challenges. Previous work on network-wide measurement mostly studied the problem of placing monitors at

proper locations to cover all measurement task (routing paths) using as few monitors as possible [20, 67, 72]. Suh et al. [72] first defined utility functions for the sampled traffic and maximized the overall utility with bounded measurement operation/deployment cost. They proposed a two phase approach where they first identified the links that should be monitored and then run a optimization algorithm to set the sampling rates. Cantieni et al. in [15] argue that most ISPs already deploy routers which are equipped with monitoring capabilities (e.g., Netflow [7], Openflow [59]) and these monitoring tools can give greater visibility on the network-wide traffic. Network operators hence can decide whether to turn on these capabilities, and there are potentially hundreds of monitoring points to choose from to achieve network-wide measurements. Based on this assumption, they reformulate the placement problem to decide which monitors should be activated and what sampling rate should be adjusted to achieve a given measurement task with high accuracy and low resource consumption. It performs more rigorous analysis on the convergence of heuristic solutions.

Upon this assumption, our design of LEISURE as a centralized network-wide measurement framework is also encouraged by recent trends in network management. [10, 14] suggest that a centralized network management approach can significantly reduce management complexity and operating costs. [66] showcases that a centralized system that coordinates monitoring responsibilities across multiple routers can significantly increase the flow monitoring capabilities of a network. The global measurement coverage can therefore be improved. In contrast, LEISURE assumes the measurement task can be fulfilled by a given set of numerous monitoring points, and its goal is to optimize the load-balancing objectives by determining which available monitors should participate in each specific measurement task and how much they need to measure instead of solving only coverage optimization problem. Also none of the previous work ever considered possible large measurement traffic, multiple measurement tasks with different costs and differentiation in the importance of traffic sub-populations, let alone load balancing among distributed monitors.



(a) Measurement task

(b) Measured by Ingress routers

(c) Measured by Egress routers

(d) Uniform load-balancing

(e) Weighted load-balancing

(f) Optimal load-balancing

Figure 3.1: Different load-balancing approaches for our toy example, which includes three OD-pair traffic as our measurement task (i.e., SF→NY, LA→Seattle, and Chicago→Atlanta, each with 120 units of traffic).

3.3 Motivating Example

We first consider the toy example with traffic demands from three OD-pairs: SF→NY, LA→Seattle, and Chicago→Atlanta, each with 120 units of traffic (IP flows) in Figure 3.1. Suppose the measurement task imposed by the network operator is to measure all the traffic from these three OD-pairs, one naive approach is to simply always measure the traffic for each OD-pair at the ingress router as shown in Figure 3.1(b). The monitors then only need to be placed in SF, LA, and Chicago with measurement traffic as 120 units. Similar to this approach, the traffic for each OD-pair can be measured at the egress router as Figure 3.1(c) shown. The monitors instead need to be placed in NY, Seattle, and Atlanta with the same measurement traffic. Both of these approaches only need 3 monitors to accomplish the assigned measurement task but with 120 unit measurement traffic.

On the other hand, assume all of these routers are equipped with monitors that are capable of performing the measurement task, our goal is to reduce their maximum measurement traffic by determining a *fraction* of the required measurement traffic to each of these monitors. One simple strategy is to *uniformly* distribute the required measurement traffic of each OD-pair to the monitors along its routing path as depicted in Figure 3.1(d). For example, the 120 units of traffic for SF→NY is measured uniformly across monitors placed in SF, Denver, Kansas City, Indianapolis and NY. Each of them takes the measurement responsibility as 24 units. Similarly, the monitors in LA, Denver, Seattle and Chicago, Indianapolis, Atlanta take the measurement responsibility as 40 units for LA→Seattle, and Chicago→Atlanta traffic respectively. The maximum measurement traffic therefore is most likely be the router with the largest number of OD-pairs passing through it (e.g., 64 units of measurement traffic in Denver/Indianapolis).

The other intuitive method distributes the required measurement traffic of each OD-pair to the monitors inverse-proportion-to the traffic passing through them as shown in Figure 3.1(e). For example, the traffic passing through SF, Denver, Kansas City, Indianapolis and NY is 120, 240, 120, 240 and 120 respectively. Based on its calculation, SF, Kansas City and NY should measure 30 units of traffic for SF→NY ($\frac{120^{-1}}{120^{-1}+240^{-1}+120^{-1}+240^{-1}+120^{-1}} \times 120$) while Denver and Indianapolis is 15 units. Similarly, the monitors in LA, Seattle, Chicago, Atlanta and Denver, Indianapolis

should take the measurement responsibility as 48, 24 units for the traffic LA→Seattle, and Chicago→Atlanta respectively.

Although these two methods achieve significant reduction in the maximum measurement traffic compared to the naive approaches (e.g., 120→64,120→48), it actually can be further reduced to 40 units as shown in Figure 3.1(f) by using LEISURE to solve the global load-balancing optimization problem. In this optimal solution, the SF→NY traffic is measured uniformly by only *three* monitors (SF, Kansas City, and NY) instead of five, each with 40 units of traffic while Denver and Indianapolis are *not* involved in the measurement of the SF→NY traffic. This in turn allows the equal splitting of the LA→Seattle traffic and the Chicago→Atlanta traffic across all three routers in each of its respective path, which results in all monitors having the same perfectly load-balanced measurement traffic as 40 units.

It is important to see that the routing path for each OD-pair traffic must overlap, such that the shared monitors can be best utilized by LEISURE to optimally minimize their maximum measurement traffic. If the monitors for measuring each OD-pair traffic are disjoint, there is no opportunity for LEISURE to globally coordinate the overall measurement task since it can only balance the monitors for each OD-pair traffic separately. Therefore the performance of LEISURE in this case degrades as the simple uniform assignments. Next, we are in general interested in finding globally optimal load-balancing solutions by using LEISURE under different network conditions (e.g., topology, traffic demand, routing matrix, etc), measurement objectives (e.g., minimize maximum workload, maximize measurement utility, etc), and resource constraints (e.g., subset of routers are capable of monitoring, some monitors have lower capacities, etc).

3.4 Leisure framework

We now present a load-balanced optimization framework to cover network-wide traffic monitoring objectives while respecting router resource constraints. ISPs typically specify their network-wide measurement task in terms of OD-pairs. To cover these measurement assignments, LEISURE needs both the traffic demand and routing information, which are readily available to network operators in [76]. In general,

LEISURE is a centralized architecture to allocate disjoint sets of required-measurement flows in OD-pairs for each router by given global network-wide information: a) network topology, monitoring infrastructure deployment, b) traffic demand, routing matrix and c) measurement requirements and the associated cost for each measurement task. The problem formulation builds up from the simplest case in which we assume: 1) the traffic matrix and routing information for the network are given exactly and they change infrequently; 2) all the routers are deployed with monitors and capable of measurement; 3) flows of each OD-pair follow a single router-level path by OSPF; and 4) there is only one measurement task for every monitor. These constraints are gradually relaxed in Section 3.5.

3.4.1 Basic Model

Let *G(V,E)* represent our network topology, where *V* is the set of routers (monitors) and *E* is the set of directed links. Each router V_i ($i = 1 \ldots M$) has two factors to limit its measurement ability: memory and bandwidth. We abstract them into a single resource constraint C_{v_i} ($i = 1 \ldots M$), the number of flows router V_i can measure in a given measurement interval.

An OD-pair, OD_x, represents a set of flows between the same pair of ingress/egress routers for which an aggregated routing placement is given. The set of all $|V| \times |V-1|$ OD-pairs is given by Θ: $OD_x, x \in \Theta$. Φ_x characterizes the traffic demand (IP flows) of the OD-pair $OD_x, x \in \Theta$ in a given measurement interval (e.g., 5 minutes). P_x represents the given routing strategy (router-level path) for every OD-pair $OD_x, x \in \Theta$.

a_x denotes the desired coverage fraction of IP flows of OD_x that is required to measure, which is imposed by the network operator. Therefore the total required measurement traffic (number of flows), β, introduced to all routers is simply a summation of traffic demand per OD-pair times a_x as $\beta = \sum_{x \in \Theta} \Phi_x \times a_x$.

Let d_i^x denote the fraction of traffic demand (IP flows) of OD_x that router V_i samples/measures (i.e., $d_i^x = \frac{\text{measured flows in } \Phi_x}{\Phi_x}$) while L_i denotes the total traffic (number of IP flows) that router V_i measures for all OD-pairs, $OD_x, x \in \Theta$ normalized by β. The

Table 3.1: Notations

Notation	Description
OD_x	represent a set of flows between the same pair of ingress/egress routers
Θ	the set of all $\|V\| \times \|V-1\|$ OD-pairs: $OD_x, x \in \Theta$
Φ_x	characterizes the traffic demand (IP flows) of OD-pair $OD_x, x \in \Theta$
P_x	represents the given routing strategy for OD-pair $OD_x, x \in \Theta$
a_x	the fraction of Φ_x (IP flows) of OD_x that is required to measure
d_i^x	the fraction of Φ_x (IP flows) of OD_x that router V_i measures
β	the total required measurement traffic (number of IP flows)
L_i	the total traffic (number of IP flows) that V_i measured normalized by β
α	load-balancing objective

summation of L_i for all routers V_i ($i = 1 \ldots M$) then equals 1. We have:

$$\beta = \sum_{x \in \Theta} \Phi_x \times a_x \tag{3.1}$$

$$L_i = \frac{1}{\beta} \sum_{x: V_i \in P_x} d_i^x \times \Phi_x \quad \forall i \tag{3.2}$$

$$\sum_{i=1}^{M} L_i = 1 \tag{3.3}$$

Our decision variable is d_i^x. The first constraint of d_i^x is that the value of d_i^x is bounded between 0 and 1 as Equation (3.4). The second constraint is that the summation of d_i^x along the path P_i for each OD-pair $OD_x, x \in \Theta$ is a_x, as Equation (3.5). If router V_i is not in the routing path P_x of OD-pair $OD_x, x \in \Theta$ ($V_i \notin P_x$), d_i^x is inherently 0. The third constraint is that the measured fraction of β for each monitor V_i should not exceed its measurement ability (resource constraint) C_{v_i} as Equation (3.6). Notations are also summarized in Table I.

$$0 \leq d_i^x \leq 1 \quad \forall x, i \tag{3.4}$$

$$\sum_{i: V_i \in P_x} d_i^x = a_x \quad \forall x \in \Theta \tag{3.5}$$

$$\sum_{x: V_i \in P_x} d_i^x \times \Phi_x \leq C_{v_i} \quad \forall i \tag{3.6}$$

Table 3.2: d_i^x for each approach with the toy example shown in Figure 3.1

	d_3^1	d_4^1	d_5^1	d_6^1	d_7^1	d_1^2	d_4^2	d_8^2	d_2^3	d_6^3	d_9^3
LB(ingress)	1	0	0	0	0	0	0	1	1	0	0
LB(egress)	0	0	0	0	1	1	0	0	0	0	1
LB(uniform)	1/5	1/5	1/5	1/5	1/5	1/3	1/3	1/3	1/3	1/3	1/3
LB(weighted)	1/4	1/8	1/4	1/8	1/4	2/5	1/5	2/5	2/5	1/5	2/5
LB(optimal)	1/3	0	1/3	0	1/3	1/3	1/3	1/3	1/3	1/3	1/3

	$MAX(L_i)$	$VAR(L_i)$	# of monitors	Decision
LB(ingress)	120/360	0.025	3	local
LB(egress)	120/360	0.025	3	local
LB(uniform)	64/360	0.00167	9	local
LB(weighted)	48/360	0.000484	9	global
LB(optimal)	40/360	0	9	global

3.4.2 Problem Formulation

We define our load-balancing objective in abstract form α, which can be any term as long as it captures load-balancing performance (i.e., identical workload for all monitors). The overall optimization objective of LEISURE is to minimize α that each router operates within its resource constraint by given parameter a_x, the required fractional coverage per OD-pair imposed by the network operator. In this section, we formulate and study three different optimization problems that correspond to three different load-balancing objective α: *min-VAR*, *min-MAX* and *min-VAR-given-MAX*.

Minimize Variance Problem (min-VAR)

In this problem, we denote α as the variance of L_i across all participating routers[1]. The intuition is that with more even workload L_i for all routers, the variance is smaller (e.g., variance=0 stands for ideal load-balancing objective where $L_i = \frac{1}{M}$ for

[1]We use “population variance" instead of “sample variance" as our objective function since we already know the number of monitors m.

all M routers). We have:

$$\alpha = VAR(L_i) = \frac{\sum_{i=1}^{M}(L_i - \bar{L})^2}{M} \quad (3.7)$$

$$\bar{L} = \frac{1}{M}\sum_{i=1}^{M} L_i = \frac{1}{M} \cdot 1 \quad (3.8)$$

This optimization problem is formulated as:

minimize α=$VAR(L_i)$

subject to

$$\frac{1}{\beta}\sum_{x:V_i \in P_x} d_i^x \times \Phi_x = L_i \qquad \forall i \quad (3.9)$$

$$\sum_{i:V_i \in P_x} d_i^x = a_x \qquad \forall x \in \Theta \quad (3.10)$$

$$\sum_{x:V_i \in P_x} d_i^x \times \Phi_x \leq C_{v_i} \qquad \forall i \quad (3.11)$$

$$0 \leq d_i^x \leq 1 \qquad \forall x, i \quad (3.12)$$

Minimize Maximum Problem (min-MAX)

In this problem, we denote α as the maximum value of L_i across all routers:

$$\alpha = MAX(L_i) \qquad i = 1 \ldots M \quad (3.13)$$

The intuition is that when LEISURE keeps minimizing the maximum value of L_i for all monitors by adjusting decision variables d_i^x, other smaller L_i will increase, eventually they will reach some equilibrium state that no more adjustments it can do to lower the $MAX(L_i)$ without increasing other L_i above $MAX(L_i)$. The problem formulation shares the same constraints as min-VAR problem, Equation (3.9) to (3.12), except that the objective function is different: minimize $\alpha = MAX(L_i), i = 1 \ldots M$.

Minimize Variance with Max-Constraint Problem (min-VAR-given-MAX)

This problem involves two phases. In the first step, we formulate the *min-MAX* problem given in Section 3.4.2 to find the minimum achievable maximum value L_{max}

(L_{max}= minimized $MAX(L_i), i = 1 \ldots M$) for all routers to cover the total required-measurement IP flows, β. Then we seek for any opportunity to further re-distribute the measurement task (workload) evenly within this constraint. Therefore in the second step, we introduce additional constraints to the *min-VAR* problem given in Section 3.4.2 to limit the L_i for each router V_i to be at most L_{max}. We then minimize the variance of L_i across all routers. Specifically, we only need to introduce the following constraint to the *min-VAR* problem:

$$L_i = \frac{1}{\beta} \sum_{x:V_i \in P_x} d_i^x \times \Phi_x \leq L_{max} \qquad \forall i \tag{3.14}$$

Therefore the *min-VAR-given-MAX* problem actually combines the *min-VAR* and *min-MAX* problems.

3.4.3 Optimal/Heuristic Solutions

We seek for the optimal d_i^x assignments for the above three problems. There is a variety of optimization tools that we can leverage. Specifically, the optimal solutions can be found by using a Quadratic Programming (QP) formulation for the *min-VAR* problem and a Linear Programming (LP) formulation for the *min-MAX* problem. The combined problem, *min-VAR-given-MAX*, can be solved in a two-phase manner by using LP first and QP follows. We refer these three optimal solutions of LEISURE as LB(min-VAR), LB(min-MAX), and LB(min-VAR-given-MAX), respectively.

Besides the optimal solutions, we introduce one simple heuristic method called LB(weighted) under the assumption that routers can always fulfill assigned measured tasks (e.g., no resource constraints for all routers in Equation (3.6)). LB(weighted) calculates d_i^x in inverse-proportion to the total required-measurement traffic amount (IP flows) passing through router V_i. The rationale behind it is that routers with larger required-measurement IP flows passing through should be assigned with less IP flows to measure in order to achieve load-balancing objective. Let β_i denote the total required measurement traffic passing through router V_i, which can be calculated using

Equation (3.15). The d_i^x assignment for LB(weighted) is formulated as:

$$\beta_i = \sum_{x:V_i \in P_x} \Phi_x \cdot a_x \qquad \forall i \in V \tag{3.15}$$

$$d_i^x = \frac{\frac{1}{\beta_i}}{\sum_{i:V_i \in P_x} \frac{1}{\beta_i}} \times a_x \qquad \forall x, i \tag{3.16}$$

Although LB(weighted) does not necessarily lead to the optimal solution, its computation time is very fast comapred to the time required to solve QP or LP optimization problems for LB(min-VAR), LB(min-MAX), and LB(min-VAR-given-MAX). In Section 3.6, we compare their load-balacning performances also with the following three simple naive strategies:

1. LB(ingress): the required measurement traffic, $\Phi_x \cdot a_x$ for each OD-pair $OD_x, x \in \Theta$, is only measured at ingress routers.

2. LB(egress): the required measurement traffic, $\Phi_x \cdot a_x$ for each OD-pair $OD_x, x \in \Theta$, is only measured only at egress routers.

3. LB(uniform): the required measurement traffic, $\Phi_x \cdot a_x$ for each OD-pair $OD_x, x \in \Theta$, is measured evenly across the routers on its routing path P_x.

Table 3.2 summarizes the corresponding d_i^x for each approach with the toy example presented in Figure 3.1. In this example, LB(min-VAR), LB(min-MAX), and LB(min-VAR-given-MAX) all have the same optimal load-balancing performance (i.e., $MAX(L_i) = \frac{40}{360}$ and $VAR(L_i) = 0$), which we denote as LB(optimal). In comparison, LB(ingress) and LB(egress) have poorest load-balancing performance but with least number of deployed monitors. LB(uniform) outperforms them but needs more monitors (e.g., 9 instead of 3 monitors in our toy example). LB(weighted) and LB(optimal) which consider global required measurement traffic can have better load-balancing performance compared to the local approaches (e.g., LB(ingress), LB(egress) and LB(uniform)), where LB(optimal) has the optimal load-balancing performance but needs much more computation time.

3.4.4 Implementation of LEISURE

The disjoint sets of required-measurement flows for each router in LEISURE could be implemented by using *hash-based packet selection* in [66] as CSAMP used, a router-level primitive suggested in Trajectory Sampling [26]. Trajectory Sampling assigns all routers in the network a common hash range and each router in the network records the passage for all packets that fall in this common hash range for applications such as fault diagnosis. In contrast, we use hash-based packet selection to assign disjoint hash ranges across multiple routers to ensure the non-overlapping measurement of traffic among monitors as CSAMP. The implementation cost of hash-based packet selection in routers could be found in [66]. Note that both LEISURE and CSAMP use the same hash-based coordination between monitors to implement disjointed flow-measurement. However, our disjoint sets of required-measurement flows for each router are the optimal result which distributes traffic measurement tasks evenly across coordinated routers while in CSAMP, their disjoint flow sets are derived from the output of an optimization framework which aims to maximize the flow-coverage objectives.

3.5 Extensions

In this section, we extend previous formulations to cover some practical scenarios, including 1) only a subset of fixed routers are deployed with monitors and capable of measurement; 2) these limited monitors are flexible to deploy in any subset of routers 3) traffic from each OD-pair follows multiple paths (e.g., ECMP: equal cost multiple path); and 4) multiple measurement tasks with different measurement costs and importance factors.

3.5.1 Measurement with Limited Monitors Scenario

In practice, not every router is equipped with monitor and capable of measurement. Suppose only K out of the M routers are deployed with monitors and thus have measurement capability. We assume each OD-pair $OD_x, x \in \Theta$ has at least one router on its routing path P_x which is capable of measurement to fulfill the measurement

tasks imposed by the network operator. Our formulation includes two problems: 1) measurement with fixed monitor deployment problem and 2) measurement with flexible monitor deployment problem.

Fixed Monitor Deployment Problem

In the first case, we assume that these K monitors have been deployed in routers and fixed. Our goal is to distribute required measurement tasks to these limited K routers. It can be simply solved by changing the routing index P_x as follows: we exclude router V_i from P_x if V_i is not equipped with monitor and unable to measure (e.g., $P_x^* = P_x - \{V_i\}$ for all OD-pair $OD_x, x \in \Theta$). Variance calculation should also be modified accordingly since we now only have K monitors instead of M. All constraints remain the same except that P_x are replaced by P_x^* in Equation (3.17)-(3.19).

$$P_x^* = P_x - \{V_i\}, \text{if } V_i \text{ is not deployed with monitor} \tag{3.17}$$

$$VAR(L_i) = \frac{\sum_{i=1}^{K}(L_i - \bar{L})^2}{K} \tag{3.18}$$

$$\bar{L} = \frac{1}{K}\sum_{i=1}^{K} L_i = \frac{1}{K} \tag{3.19}$$

Flexible Monitor Deployment Problem

In the second case, the location of K monitors have not been decided and they are flexible to be deployed in any router. This problem includes not only the *distribution* of measurement tasks, but also the *placement* of monitors. To formulate this problem, we introduce additional decision variables u_i, where $u_i = 1$ if router V_i is selected to deploy a monitor, and $u_i = 0$ otherwise. The summation of u_i is therefore to be K. We assume every monitor has identical limited measurement capability (resource constraint) as C_m. The problem is formulated below with load-balancing objective as $\alpha = MAX(L_i)$. Note that it is no longer an LP/QP problem since $u_i, i \in V$ are Boolean

variables.

$$\text{minimize } \alpha$$

$$\text{subject to}$$

$$\frac{1}{\beta}\sum_{x:V_i\in P_x} d_i^x \times \Phi_x \times u_i = L_i \qquad \forall i \tag{3.20}$$

$$\sum_{i:V_i\in P_x} d_i^x \times u_i = a_x \qquad \forall x \in \Theta \tag{3.21}$$

$$\sum_{x:V_i\in P_x} d_i^x \times \Phi_x \times u_i \leq C_m \qquad \forall i \tag{3.22}$$

$$\sum_{i=1}^{M} u_i = K \tag{3.23}$$

$$0 \leq d_i^x \leq 1 \qquad \forall x, i \tag{3.24}$$

$$u_i \in \{0, 1\} \qquad \forall i \tag{3.25}$$

In this model, L_i is the summation of the product of Φ_x, d_i^x and u_i. Therefore the objective function α is related to the product of two decision variables u_i and d_i^x, and the optimization problem falls into the MIQP (Mix Integer Quadratic Programming) category. In order to avoid quadratic programming, we could introduce z_i^x to decouple $d_i^x \times u_i$ by using Equations (3.26) to (3.28). It is easy to see their equivalence. When $u_i = 0$, $z_i^x = 0$ from (3.27); and when $u_i = 1$, $z_i^x = d_i^x$ from (3.28).

$$z_{ij}^y = \gamma_{ij}^y \times u_{ij} \tag{3.26}$$

$$0 \leq z_{ij}^y \leq u_{ij} \tag{3.27}$$

$$\gamma_{ij}^y + u_{ij} - 1 \leq z_{ij}^y \leq \gamma_{ij}^y \tag{3.28}$$

Although we could reduce the MIQP problem to the MILP (Mix Integer Linear Programming) problem by introducing z_i^x , the new MILP problem actually has doubled number of decision variables. This is because the cardinality of $u_i \ll$ the cardinality of d_i^x in practice. Fortunately, the decision variable d_i^x (for distributing measurement tasks) is highly dependent on the decision variable u_i (for monitor placement). If $u_i = 0$ (i.e., router V_i is not selected to deploy a monitor), router V_i cannot participate in any specific measurement task. It means d_i^x, the fraction of Φ_x (IP flows) for each OD-pair,

$OD_x, x \in \Theta$ that router V_i measures should be zero (i.e., $d_i^x = 0$). On the other hand when $u_i = 1$ (i.e., router V_i is selected to deploy a monitor and capable of measurement), d_i^x could be any decimal bounded between 0 and 1. Therefore we can directly use d_i^x to substitute $d_i^x \times u_i$ to avoid quadratic programming but with a new constraint, $0 \leq d_i^x \leq u_i$. It is easy to see their equivalence. The formulation now becomes MILP and keeps the original number of decision variables:

$$\text{minimize } \alpha$$

$$\text{subject to}$$

$$\frac{1}{\beta} \sum_{x:V_i \in P_x} d_i^x \times \Phi_x = L_i \qquad \forall i \tag{3.29}$$

$$\sum_{i:V_i \in P_x} d_i^x = a_x \qquad \forall x \in \Theta \tag{3.30}$$

$$\sum_{x:V_i \in P_x} d_i^x \times \Phi_x \leq C_m \qquad \forall i \tag{3.31}$$

$$\sum_{i=1}^{M} u_i = K \tag{3.32}$$

$$0 \leq d_i^x \leq u_i \qquad \forall x, i \tag{3.33}$$

$$u_i \in \{0, 1\} \qquad \forall i \tag{3.34}$$

Optimal MILP/ Heuristic Solutions

The optimal solution searches for the best d_i^x and u_i assignments for the hybrid load-balancing and placement problem under the assumption of using limited flexible K monitors instead of M to minimize the maximum measurement workload across them (e.g., minimize $MAX(L_i)$, $i = 1 \ldots K$). The simplified formulation is MILP problem since u_i is a binary decision variable and d_i^x is a continuous decision variable. There is a variety of optimization tools that we can leverage. In particular, we use an MILP solver (e.g., CPLEX [4]) to find the optimal solution. We refer to this solution as “Optimal". For small to medium size networks, the optimal load-balancing with placement solution can be readily found. However, given that the time-complexity of MILP problems are in general NP-hard, the solvers are not fast enough for large networks.

It is easy to see that the hybrid load-balancing and placement problem becomes a LP (Linear Programming) problem if the monitor placement strategy is given (i.e., with fixed u_{ij}). Therefore, all of our proposed heuristic solutions tend to decide the monitor locations first. In this section, we propose two heuristic solutions to approximate the optimal performance: “LB-Successive Selection" and “LB-Greedy". Both of them iteratively select monitors to disable, based on the different planned monitor placement strategy decided from the previous iteration. They all start from an initial configuration under the assumption that all M routers are fully deployed with monitors. We refer to this initial configuration as the “All-On" stage. The monitor-disable process is repeated until only K out of M monitors are left.

Algorithm 4 LB-Successive Selection Algorithm

```
1:  while More than K monitors are left do
2:      Minimize α by using all remaining monitors
3:      find the corresponding d_i^x
4:      for Each remaining monitor V_i ∈ M̂ do
5:          Calculate L_i for each remaining monitor with d_i^x
6:          Calculate one of the three metrics with d_i^x
7:      end for
8:      Find monitor with minimum L_i
9:      if only one monitor has minimum L_i then
10:         Disable that monitor
11:     else
12:         Disable the monitor with minimum L_i and least performance-metric
13:     end if
14: end while
```

LB-Successive Selection: it starts from the initial All-On configuration where all M routers are assumed to be fully deployed with monitors, and iteratively chooses one monitor to disable after optimization process (i.e., minimize $MAX(L_i)$) until only K out of M monitors are left. The selection of which monitor to disable is based on their ranking of measurement workload (e.g., L_i). We choose the one having least measurement workload across all remaining monitors (e.g.,V_i=$\min(L_i)$,

i=1 . . . $\hat{M}$), where $\hat{M}$ stands for the set of remaining routers deployed with monitors. The intuition is that the monitors with higher measurement workloads after optimization process (i.e., minimize MAX(L_i),$\forall i$, the maximum workload across all monitors) take more measurement responsibility for the traffic from some OD-pairs which have few monitors deployed in their routing paths. Therefore, those monitors can not be disabled, otherwise their assigned measurement task can not be further redistributed. If more than two routers have the same minimum measurement workload in each iteration, LB-Successive Selection calculates one of the following three metrics which are served as tie-breaker and disables the one with least value:

- *Least-traffic ($\sum_{x:V_i \in P_x} \Phi_x$).* The intuition is that the monitors with the least amount of traffic passing through them have less freedom to load-balance the measurement tasks for each OD-pair traffic.

- *Least-LB(uniform).* We use LB(uniform) heuristic mentioned in Section 3.4.3 to find corresponding measurement workload across all remaining monitors to serve as our second-stage tie-breaker.

- *Least-LB(weighted).* We use LB(weighted) heuristic to find corresponding measurement workload across all remaining monitors to serve as our second-stage tie-breaker.

In particular, it disables monitor based on their ranking calculated from the previous iteration (Line 12). This means we use the information from the previous iteration (i.e., planned measurement fraction d_i^x) to calculate the metric for each monitor in the current iteration (Line 5-6).

LB-Greedy Algorithm: similar to LB-Successive Selection, the LB-Greedy algorithm also disables one monitor in each iteration, until K monitors are left. However, it is more time-consuming since it tests all remaining monitors one-by-one in each iteration. To test the importance of each monitor, LB-Greedy re-computes the minimized α after turning off each monitor alternately (Line 2-7), which essentially involves numerous optimization procedure (Line 4) mentioned in Section 3.4.3. Based on the testing of every remaining monitor, it disables the one that has least impact on α (Line 8).

Since the LB-Greedy algorithm exhaustively tests individual monitor in each iteration, its performance is expected to be close to the optimal MILP solution. However it is still suboptimal since LB-Greedy only tests individual monitor instead of every possible combination. Besides, the algorithm remains computationally costly, since it tests $O(\hat{M})$ monitors with $O(\hat{M})$ LP problems in each iteration. For a moderate sized topology, an MILP solver can sometimes work faster than this LB-Greedy approach. Details are shown in Section 3.6.3.

Algorithm 5 LB-Greedy Algorithm

while More than K monitors are left **do**
  for Each remaining monitor $V_i \in \hat{M}$ **do**
    Disable the monitor, V_i
    Minimize α based on remaining monitors
    Store α
    Enable the monitor, V_i
  end for
  Find monitor, V_i, with smallest $\alpha \in \hat{M}$ when they are disabled
  $\hat{M} \leftarrow \hat{M}/\{V_i\}$
end while

3.5.2 Multi-Path Routing Scenario

All the sections above have assumed single-path routing (e.g., OSPF). In this section, we extend our work to support “load-balancing" of measurement tasks in the case of multi-path routing (e.g., ECMP). Since ECMP enables routers to make forwarding decisions on a per IP-flow basis rather than on a per-packet basis, packets for a single flow will still follow one path.

Our formulation treats each of the different paths as a distinct virtual OD-pair with different portions of the origin traffic demand. Suppose each OD-pair $OD_x, x \in \Theta$ has N_x routing paths, denoted as P_{xh} ($h = 1 \ldots N_x$) with total traffic demand Φ_x. We create virtual OD-pairs OD_{xh} for each path P_{xh} ($h = 1 \ldots N_x$) of OD-pair $OD_x, x \in \Theta$ with traffic demand Φ_{xh} where $\sum_{h=1}^{N_x} \Phi_{xh} = \Phi_x, \forall x$. We also let a_{xh} denote the given

fraction of Φ_{xh} that is required to be measured for each virtual OD-pair OD_{xh} where $\sum_{h=1}^{N_x} a_{xh} = a_x, \forall x$. d_i^{xh} denotes the fraction of Φ_{xh} that router V_i measures for each virtual OD-pair OD_{xh}. The problem can be formulated below. In this formulation, d_i^{xh} are the decision variables. L_i and a_{xh} can in turn be calculated as functions of d_i^{xh}. α can still be defined according to different optimization criteria.

$$\text{minimize } \alpha$$

$$\text{subject to}$$

$$\sum_{x\in\Theta}\sum_{h=1}^{N_x} \Phi_{xh} \times a_{xh} = \beta \tag{3.35}$$

$$\frac{1}{\beta}\sum_{x:V_i\in P_{xh}}\sum_{h=1}^{N_x} d_i^{xh} \times \Phi_{xh} = L_i \qquad \forall i \tag{3.36}$$

$$\sum_{i:V_i\in P_{xh}} d_i^{xh} = a_{xh} \qquad \forall h, x \tag{3.37}$$

$$\sum_{x:V_i\in P_{xh}}\sum_{h=1}^{N_x} d_i^{xh} \times \Phi_{xh} \leq C_{v_i} \qquad \forall i \tag{3.38}$$

$$0 \leq d_i^{xh} \leq 1 \qquad \forall h, x, i \tag{3.39}$$

3.5.3 Measurement with Multiple Tasks Scenario

Until now, we have assumed a single measurement task/function with identical unit cost at every router. In practice, traffic measurement may involve multiple tasks with different measurement cost factors (e.g., DPI is much more resource-intensive than say counting). It is important that we evenly distribute measurement tasks to monitors in this setting. Meanwhile, in some fringe cases, different measurements might compete for limited resources. It is also important to study how they cooperate to achieve better global measurement.

Therefore we have two optimization objectives: 1) minimize the maximum value of L_i for all routers ($i = 1 \ldots M$) from load-balancing perspective; 2) maximize the aggregated measurement utility across all measurement tasks. This joint optimization problem involves two phases. In the first step, we use the *min-MAX* problem formulation given in Section 3.4.2 to find the minimum achievable maximum value L_{max} to fulfill

every requested measurement task for all routers by temporarily ignoring routers' measurement capabilities (resource constraints). In the second step, we introduce θ_i to reflect the resource constraints for all routers by limiting their L_i to not exceed $\theta_i \times L_{max}$ as $L_i \leq \theta_i \times L_{max}, i = 1 \ldots M$ where $0 \leq \theta_i \leq 1$. The more severe the resource constraint is (i.e., with smaller C_{V_i}), the lower the θ_i will be while $\theta_i = 1$ means no resource constraint for router V_i. We then maximize the measurement utility for all tasks under limited resource constraints and load-balancing conditions.

We assume there are in total ζ measurement tasks. Each task, denoted as t ($t = 1 \ldots \zeta$), is characterized by its measurement cost C^t. Let a_{xt} denote the given fraction of Φ_x that is required to be measured for each measurement task t ($t = 1 \ldots \zeta$) per OD-pair $OD_x, x \in \Theta$. We assume single path routing for every OD-pair $x \in \Theta$ and all routers are capable of processing every measurement task. Our first optimization problem is to evenly distribute the measurement tasks/costs across all routers where the measurement capabilities (resource constraints) of all routers are temporarily ignored. We choose the load-balancing objective as α =MAX(L_i) and the problem is formulated below.

$$\text{minimize } \alpha = \text{MAX}(L_i)$$

$$\text{subject to}$$

$$\sum_{x \in \Theta} \Phi_x \times \sum_{t=1}^{\zeta} a_{xt} \times C^t = \beta \tag{3.40}$$

$$\frac{1}{\beta} \sum_{x:V_i \in P_x} \Phi_x \times \sum_{t=1}^{\zeta} d_i^{xt} \times C^t = L_i \qquad \forall i \tag{3.41}$$

$$\sum_{i:V_i \in P_x} d_i^{xt} = a_{xt} \qquad \forall x, t \tag{3.42}$$

$$0 \leq d_i^{xt} \leq 1 \qquad \forall x, t, i \tag{3.43}$$

After the optimal minimum achievable maximum workload L_{max} is found for every router (with no resource constraint) to cover all measurement tasks (L_{max}= minimized $MAX(L_i), i = 1 \ldots M$), we next consider that routers have their own resource constraint C_{V_i} which may make their $L_i < L_{max}$ and fail partial measurement tasks. Let θ_i denote the fraction of L_{max} to reflect the resource constraint C_{V_i} for router

V_i ($i = 1 \ldots M$) as $\theta_i = \min(\frac{C_{V_i}}{L_{max} \cdot \beta}, 1)$ where $0 \leq \theta_i \leq 1$ [2] We introduce a new constraint for all routers to bound their L_i by $\theta_i \times L_{max}$ as[3]:

$$L_i \leq \theta_i \times L_{max} \qquad \forall i \tag{3.44}$$

Under this constraint, we study how different measurement tasks are assigned with proper portion of resources such that the overall measurement utility is maximized. Let I^t denote the importance factor for each measurement task t ($t = 1 \ldots \zeta$) and G denote the ideal aggregated measurement utility weighted by I^t for all measurement tasks without considering resource constraints at routers, $G = \sum_{x \in \Theta} \Phi_x \times \sum_{t=1}^{\zeta}$. G_i denotes the total measurement utility that router V_i gets for all measurement tasks normalized by G, $G_i = \frac{1}{G} \sum_{x:V_i \in P_x} \Phi_x \times \sum_{t=1}^{\zeta} d_i^{xt} \times I^t$. The optimization problem can be formulated as follows, with d_i^{xt} as the decision variables:

$$\text{maximize} \sum_{i=1}^{M} G_i$$

$$\text{subject to}$$

$$\sum_{x \in \Theta} \Phi_x \times \sum_{t=1}^{\zeta} a_{xt} \times C^t = \beta \tag{3.45}$$

$$\frac{1}{\beta} \sum_{x:V_i \in P_x} \Phi_x \times \sum_{t=1}^{\zeta} d_i^{xt} \times C^t = L_i \qquad \forall i \tag{3.46}$$

$$\theta_i \times L_{max} \geq L_i \qquad \forall i \tag{3.47}$$

$$\sum_{i:V_i \in P_x} d_i^{xt} \leq a_{xt} \qquad \forall x, t \tag{3.48}$$

$$0 \leq d_i^{xt} \leq 1 \qquad \forall x, t, i \tag{3.49}$$

The value of normalized objective function, $\sum_{i=1}^{M} G_i$, is always in the range as $0 < \sum_{i=1}^{M} G_i \leq 1$. For the case when θ_i=1 $\forall i$, $\sum_{i=1}^{M} G_i = 1$, which means all the required measurement tasks can be satisfied ($\sum_{i:V_i \in P_x} d_i^{xt}$=$a_{xt}$ $\forall x, t$) and the aggregated measurement utility is maximum since there is no resource constraints on all routers.

[2] θ_i=1 implied that there is no resource constraint on router V_i since the traffic amount it measured is less than its resource constraint: $L_i \times \beta \leq L_{max} \times \beta \leq C_{V_i}$.

[3] By substituting θ_i=$\min(\frac{C_{V_i}}{L_{max} \cdot \beta}, 1)$ with Equation 3.41 into Equation 3.44, the traffic amount, $\sum_{x:V_i \in P_x} \Phi_x \times \sum_{t=1}^{\zeta} d_i^{xt} \times C^t$, that router V_i measured is always less than its resource constraint (C_{V_i}).

However, with the resource constraints (θ_i decreases), only a subset of measurement tasks can be fulfilled, and the goal of the above formulation is to maximize the global measurement utility and maintain the load-balancing conditions simultaneously.

3.6 Performance Evaluation

We evaluated the performance of LEISURE with three optimal solutions, LB(min-VAR), LB(min-MAX) and LB(min-VAR-given-MAX), for different load-balancing objectives in various realistic scenarios on two separate real, large point-of-presence(PoP)-level backbone networks: Abilene [16] and GEANT [73]. We also compare them with several simple naive approaches, namely LB(ingress), LB(egress), LB(uniform), and LB(weighted). Our starting point is to conduct a preliminary evaluation on the basic model in Section 3.6.2 based on three assumptions: (1) all routers are equipped with monitors that are capable of performing the measurement task, (2) traffic from each OD-pair has a single router-level path by OSPF and (3) there is only one measurement task. We relax these assumptions in Section 3.6.3 and Section 3.6.4 to show our load-balancing ability and computation time complexity. Section 3.6.5 presents our load-balancing and measurement utility maximizing results for the scenario of multiple measurement tasks with different cost and importance factors.

3.6.1 Experimental Setup and Performance Metrics

We use two real datasets from the Abilene [16] and GEANT networks [73], both of which have been studied and discussed in the research literature. Their data sets are publicly available, including network topology, routing information. Based on these available data sets, we implemented a flow-based trace-driven simulation to conduct our evaluations. For both networks, we use the real traffic matrices provided by a third party [5]. The traffic matrix data sets for the Abilene network are available at [2], and the traffic matrix data sets of the GEANT network are available at [6].

Abilene: A public academic network in the U.S. with 11 nodes interconnected by OC192, 10 Gbits/s links. The traces we use were collected from April 22-26, 2004. The routers in ATLA, CHIN, DENV, HSTN, IPLS, KSCY, LOSA, NYCM, SNVA,

STTL and WASH are denoted as $R_0, R_1, \cdots, R_{10}$ respectively.

GEANT: It connects a variety of European research and education networks. Our experiments were based on the December 2004 snapshot available at [3], which consists of 23 nodes and 74 links varied from 155 Mbits/s to 10 Gbits/s. The traces we use were collected from April 11-15, 2004.

The traffic matrix we use consists of demands for every OD-pair within a certain time interval (5 mins for Abilene and 15 mins for GEANT). We construct OD-pairs by considering all possible pairs of PoPs and calculate their shortest-path routes. In brief, these traffic matrices are derived from flow information collected at key locations of the network, and is transformed into the demand rate for each OD-pair based on the control plane information.

In the following sections, we assume our target is to measure all traffic (i.e., $a_x = 1, \forall x \in \Theta$). Therefore the workload L_i for router R_i ($i = 1 \ldots M$) is defined as the traffic amount that router R_i measured normalized by the total traffic demand. Theoretically, the ideal load-balancing workload L_i for M monitors is $\frac{1}{M}$. However, it might be unachievable due to routing limitations from TE or resource constraints on monitors. In our experiments, we are interested in the following three performance metrics:

- *Maximum Workload*: We use the maximum value of each monitor's measurement workload in the entire network to serve as our load-balancing performance metric mainly (e.g., MAX(L_i),i=$1 \ldots M$).

- *Variance of Workload*: The other load-balancing performance metric used in this paper is the variance of workloads across all monitors (e.g., VAR(L_i)).

- *Computation Time*: In our experiment, we only collect computation time for the LP or MILP solver since they usually take much longer time compared to normal numerical computation, and therefore dominate the whole computation time of LEISURE. Meanwhile, the computation time for LP or MILP may vary for different solvers. We therefore do not mix them with other numerical computations.

Table 3.3: Comparisons on Maximum value of L_i

	Naive Approaches		Heuristic Approaches	
Network	ingress	egress	uniform	weighted
Abilene	19.16%	29.59%	21.67%	12.12%
GEANT	28.79%	13.19%	13.73%	10.67%

	Optimal Load-Balancing		
Network	min-VAR	min-MAX	min-VAR given MAX
Abilene	10.11%	9.45%	9.45%
GEANT	6.15%	6.06%	6.06%

Table 3.4: Comparisons on Variance of L_i

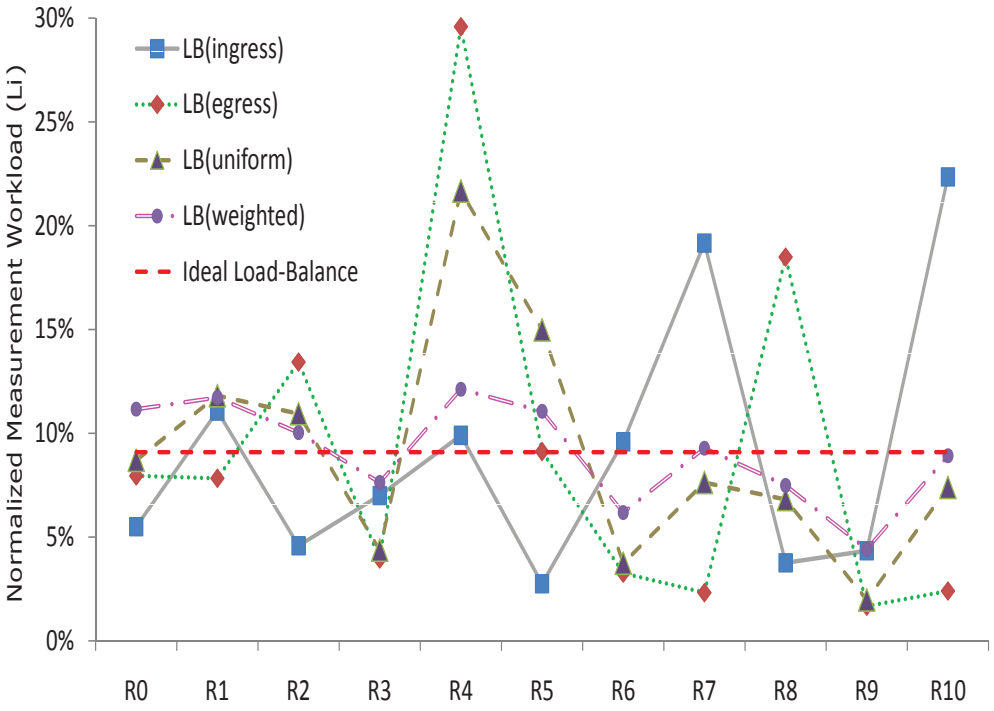
	Naive Approaches		Heuristic Approaches	
Network	ingress	egress	uniform	weighted
Abilene	0.004107	0.007366	0.003158	0.000602
GEANT	0.003978	0.001626	0.001594	0.000662

	Optimal Load-Balancing		
Network	min-VAR	min-MAX	min-VAR given MAX
Abilene	0.000105	0.000131	0.000105
GEANT	0.000378	0.000495	0.000378

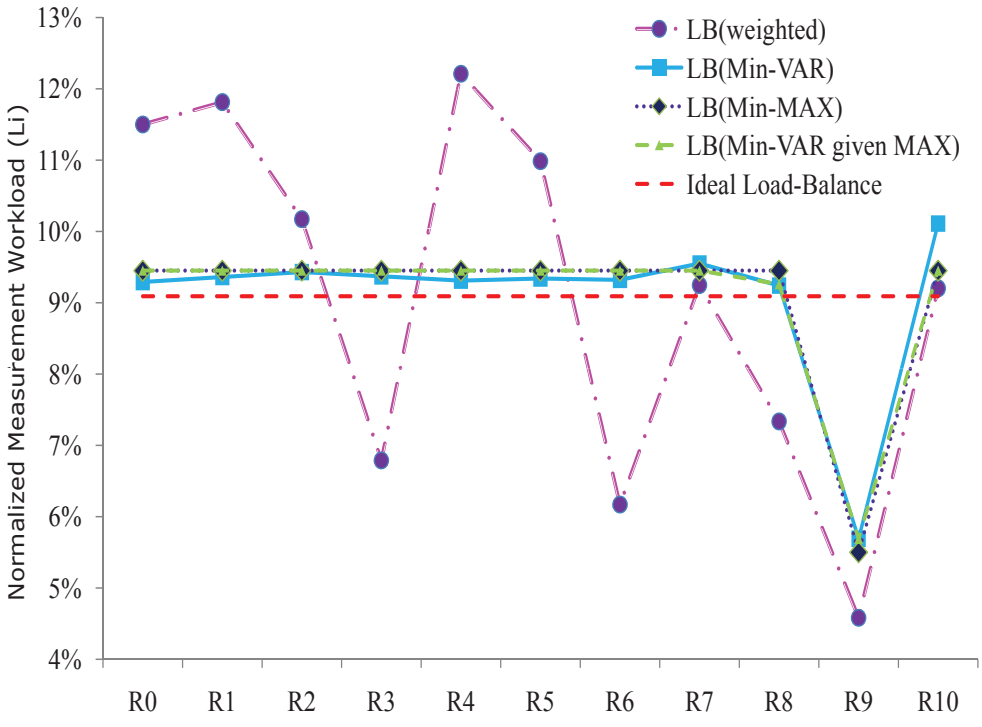
3.6.2 Basic Load-Balancing Comparison

In this section, we compare the load-balancing performance of all approaches based on two assumptions (ubiquitous monitors and single path routing). Table 3.3 compares MAX(L_i) of all monitors for different approaches. For GEANT, our optimal load-balancing solutions can reduce MAX(L_i) by a factor of 4.75X(=$\frac{28.79\%}{6.06\%}$) when compared to the naive approach of LB(ingress) and 2.27X(=$\frac{13.73\%}{6.06\%}$) when compared to LB(uniform). Similar gains can be seen in the results for Abilene as well. Figure 3.2 and Figure 3.3 plot in more details the L_i values of 11 monitors and 23 monitors for different load-balancing approaches in Abilene and GEANT networks respectively.

Another relative performance measure is to see how close the maximum workloads are in comparison to the ideal load-balancing case of $\bar{L} = \frac{1}{M}$, as given

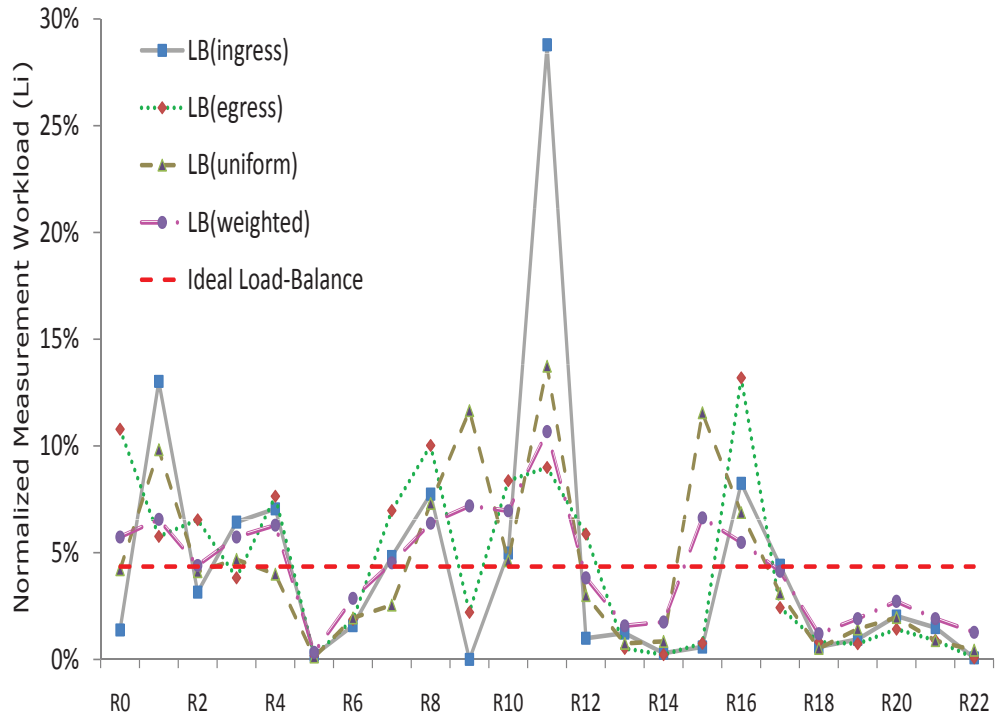


(a) Simple Heuristic Approaches in Abilene

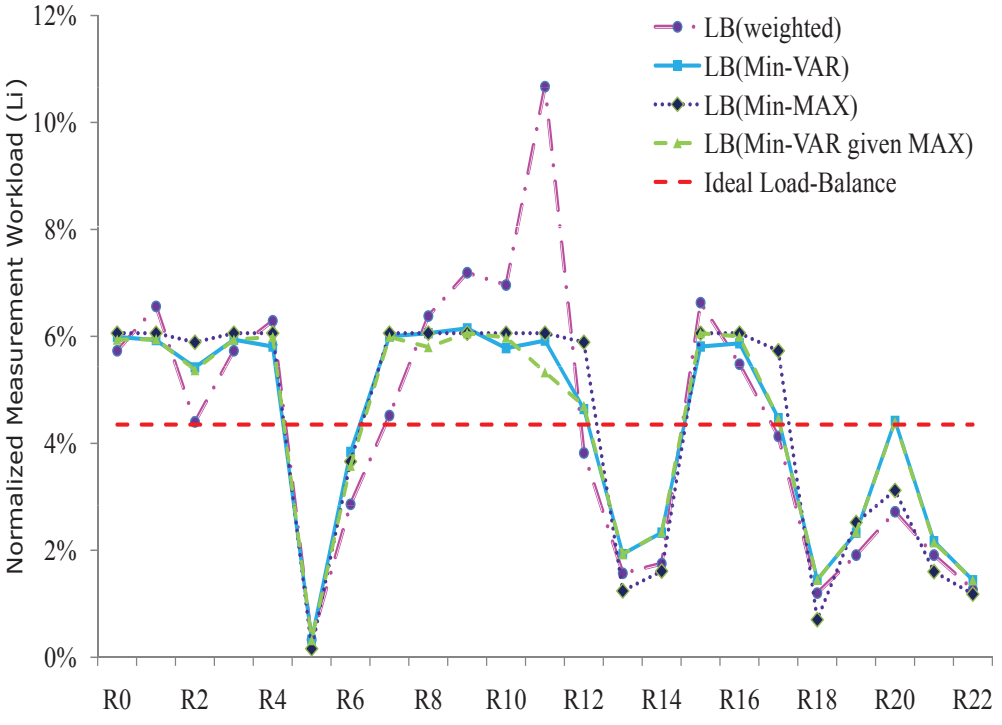


(b) Optimal Approaches in Abilene

Figure 3.2: Measurement load distribution for different approaches in Abilene



(a) Simple Heuristic Approaches in GEANT



(b) Optimal Approaches in GEANT

Figure 3.3: Measurement load distribution for different approaches in GEANT

by Eq.(3.8). For Abilene and GEANT, the ideal $\bar{L}$ is 9.09%(=$\frac{1}{11}$) and 4.35%(=$\frac{1}{23}$), respectively. However, the MAX(L_i) of LB(ingress) for Abilene and GEANT are 19.16% and 28.79%, respectively, which are 2.11X and 6.62X worse than the ideal case. For simple heuristic approaches, they still have large MAX(L_i) values compared to the ideal case: e.g., 21.67% (2.3X worse) for LB(uniform) in Abilene and 10.67% (2.4X worse) for LB(weighted) in GEANT. On the other hand, our three optimal load-balancing solutions presented in Figure 3.4 and Table 3.3 perform very close to the theoretical ideal case: 10.11%, 9.45%, and 9.45% for LB(min-VAR), LB(min-MAX), and LB(min-VAR given MAX), respectively, as compared to the ideal case of 9.09% for Abilene. Similarly, our three optimal solutions are 6.15%, 6.06%, and 6.06%, respectively, as compared to the ideal case of 4.35% for GEANT.

Table 3.4 compares VAR(L_i) across all monitors for different approaches. For Abilene, our optimal load-balancing solutions can reduce VAR(L_i) by a factor of 70X(=$\frac{0.007366}{0.000105}$) when compared to the naive approach of LB(egress), and over 30X(=$\frac{0.003158}{0.000105}$) when compared to LB(uniform). Similar improvements in variance can be seen for GEANT as well.

To better understand why our optimal solutions can achieve more evenly distributed measurement load, we use traffic from only five OD-pairs in Abilene[4] to show the detailed load assignment in Figure 3.4 (WAS-DNV, NYC-HST, DNV-IPL, CHI-LOS and ATL-STT with 66.5 MB, 44.9 MB, 44.6 MB, 19.8 MB and 11.7 MB, respectively). In Figure 3.4(a), although LB(uniform) distributes each OD-pair traffic to all monitors in the path uniformly (e.g., WAS-DNV with 6 monitors), the aggregated workload for overall measurement task in each monitor is still unbalanced (e.g., L_i for all routers R_i ($i = 1 \ldots 10$) are distributed between 1% to 17%). LB(weighted) in Figure 3.4(b) improves the load-balancing performance due to the global view it has but still load-balanced poorly (e.g., L_i distributed between 4% to 14%). In contrast, the optimal solutions can achieve much better load-balancing performance (e.g., L_i distributed between 5.5% to 10.5%) by excluding some monitors from measuring certain OD-pair traffic (e.g., R4 and R5 do not measure traffic for WAS-DNV OD-pair in Figure 3.4(d)).

[4]The notations of these OD-pairs and their routing information could be found in [16], [2].

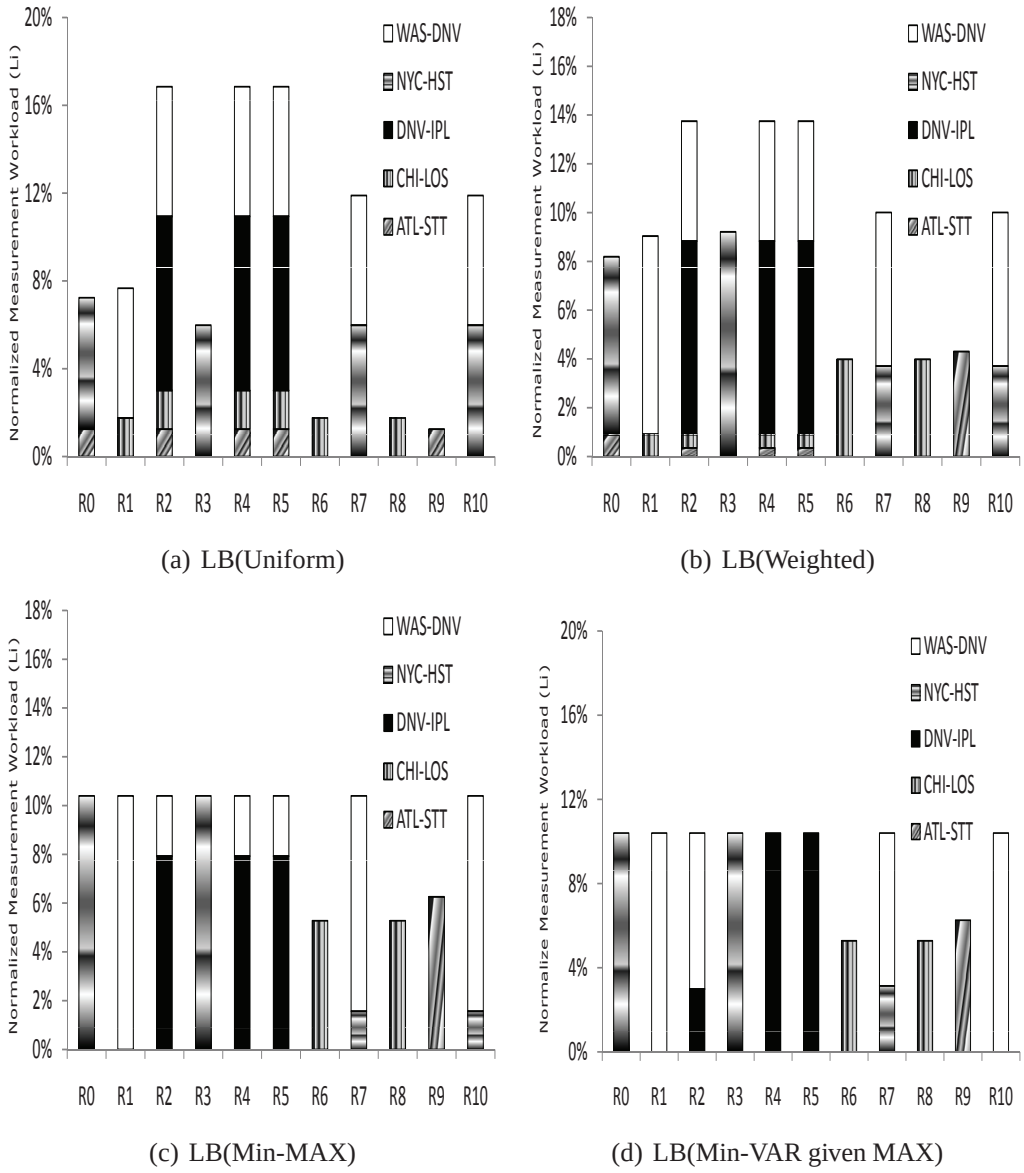


Figure 3.4: (Continued) Detailed Abilene results for five OD-pairs. Optimal solutions allow nodes to be excluded from measurement if they are already overloaded.

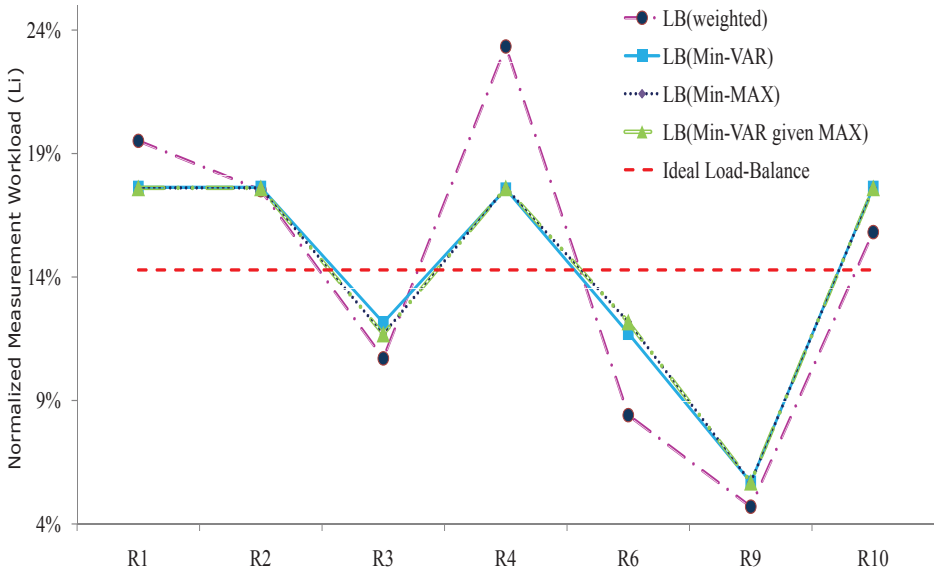


Figure 3.5: Measurement load distribution with limited 7 out of 11 monitors in Abilene.

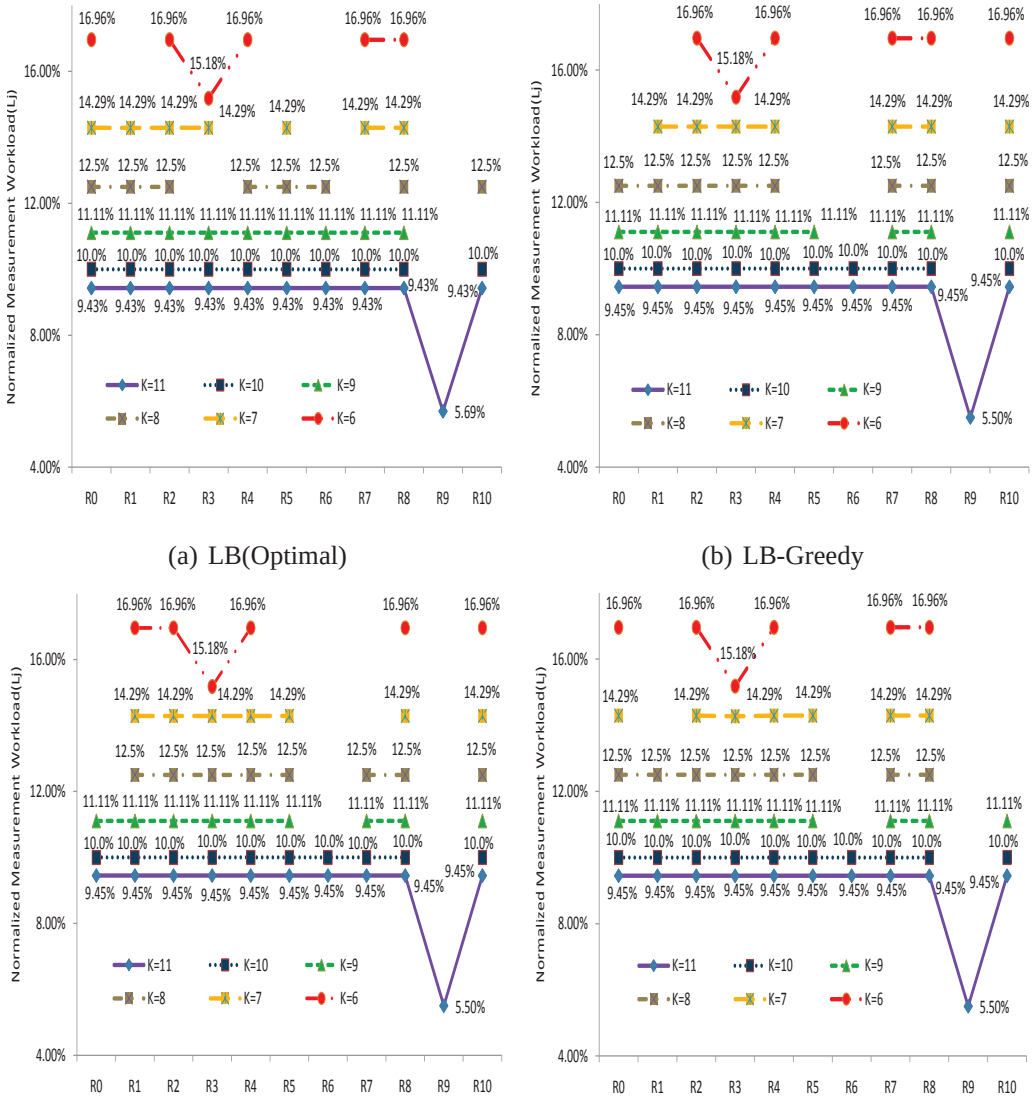
3.6.3 Limited number of Monitors

In this section, we relax our first assumption to the case that only a subset of routers are deployed monitors and capable of measurement. We further evaluate LEISURE in the following two scenarios: 1) measurement with fixed monitor deployment scenario and 2) measurement with flexible monitor deployment scenario.

Fixed Monitor Deployment Scenario

In the first case, we assume there are $K = 7$ out of $M = 11$ routers are deployed with fixed monitors in Abilene. The routers which are excluded to deployed monitors are R_0, R_5, R_7 and R_8[5]. Therefore LEISURE can only distribute the measurement task to the remaining 7 monitors. We omit naive approaches and focus on heuristic (i.e., LB(weighted)) and optimal approaches in Figure 3.5. Compared with ubiquitous case in Figure 3.2(a), the ideal load-balancing workload is increased from 9.09% to 14.29%. For LB(min-VAR), LB(min-MAX) and LB(min-VAR given MAX), the MAX(L_i) is

[5]The reason to choose those 4 excluded routers is to maintain the fact that at least one capable monitor in each OD-pair's route to fulfill the measurement tasks imposed by the network operator.



(a) LB(Optimal)

(b) LB-Greedy

(c) LB-Successive Selection(traffic)

(d) LB-Successive Selection(weighted)

Figure 3.6: Measurement load distribution for different approaches in Abilene with limited K flexible deployed monitor

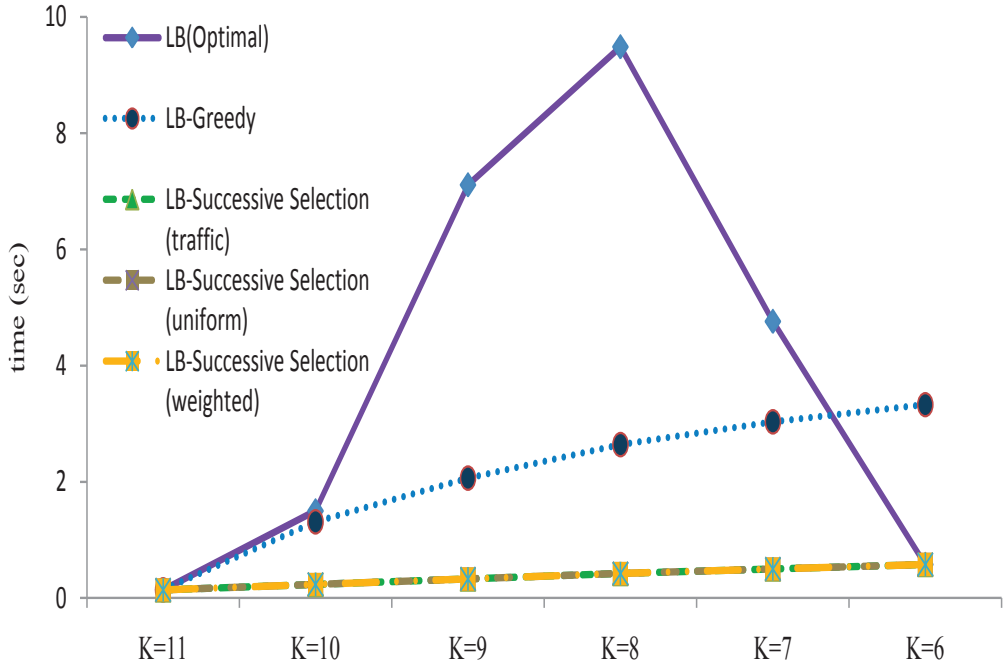
only increased from 9.67% to 17.61%. However, for heuristic approach, LB(weighted), MAX(L_i) increased from 12.12% to 23.33%. We observe that LEISURE with these three optimal solutions for different load-balancing objectives only increased 7.94% workload for MAX(L_i), which are close to 5.2% for the theoretical ideal case and are much better than 11.21% for LB(weighted).

Flexible Monitor Deployment Scenario

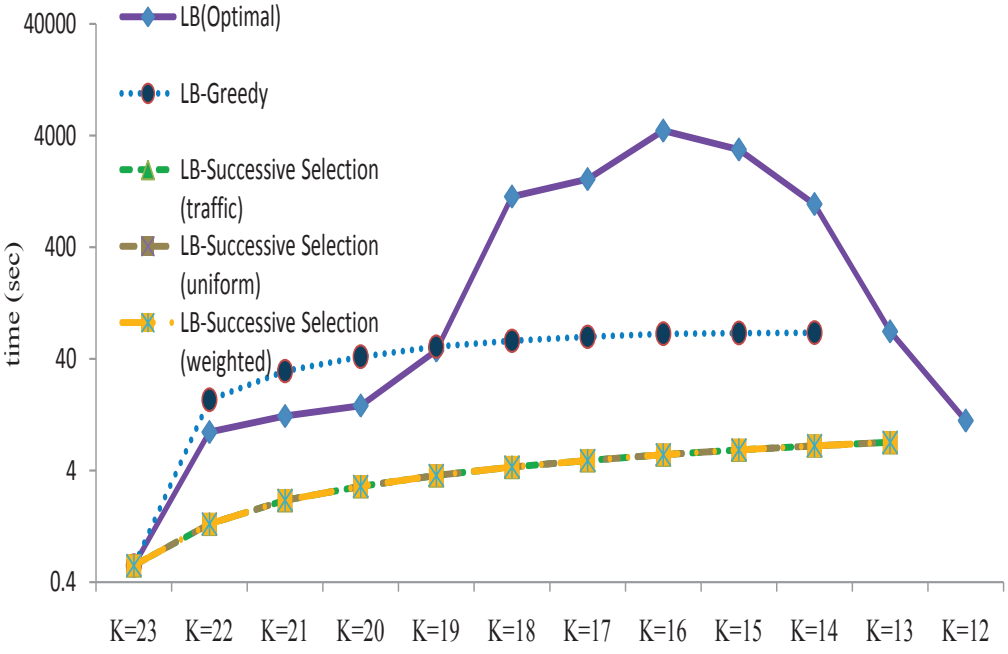
In the second case, we assume there are limited K out of $M = 11$ routers are deployed with flexible monitors in Abilene (e.g., K=11, 10, 9, 8, 7 and 6). Therefore LEISURE can only distribute the measurement task to these K monitors. Figure 3.6 plots the detailed L_i (normalized measurement workload) values of monitors for different load-balancing approaches in Abilene. Compare to the ideal load-balancing case of $\bar{L} = \frac{1}{K}$, as given by Eq.(3.8) where K is the limited number of deployed monitors, our optimal MILP solution (e.g., LB(Optimal)) performs almost the same as the ideal case. As shown in Figure 3.6(a), for K=11, 10, 9, 8, 7 and 6 in Abilene, the L_{max} (L_{max}=minimized MAX(L_i), $i = 1 \ldots M$) of LB(Optimal) is 9.43%, 10.0%, 11.11%, 12.5%, 14.29% and 16.96%, respectively while the ideal $\bar{L}$ is 9.09%(=$\frac{1}{11}$), 10.0%(=$\frac{1}{10}$), 11.11%(=$\frac{1}{9}$), 12.5%(=$\frac{1}{8}$), 14.29%(=$\frac{1}{7}$) and 16.67%(=$\frac{1}{6}$).

Due to the potentially long computation time required to solve for the LB(Optimal), we propose several heuristic algorithms to reduce the computation time complexity. They are categorized as “Greedy" and “Successive Selection". We first show that they all have nearly equivalent load-balancing performance, L_{max}, as LB(Optimal). In Figure 3.6(b), the L_{max} for K=11, 10, 9, 8, 7 and 6 in LB-Greedy is 9.45%, 10.0%, 11.11%, 12.5%, 14.29% and 16.96% respectively which is nearly the same as 9.43%, 10.0%, 11.11%, 12.5%, 14.29% and 16.96% in LB(Optimal) although their deployment of monitors might be different. The same observation could be also found in LB-Successive Selection(traffic)/LB-Successive Selection(uniform), and LB-Successive Selection(weighted) (e.g., see Figure 3.6(c) and 3.6(d)). We omit the result of LB-Successive Selection(uniform) since it performs close to LB-Successive Selection(traffic).

Next we compare their computation time complexity with LB(Optimal). As



(a) Time Complexity in Abilene



(b) Time Complexity in GEANT

Figure 3.7: Computation cost for different Approaches with various limited K flexible monitor deployment in Abilene/GEANT

shown in Figure 3.7(a), LB-Greedy reduces computation times by a factor of 3.6X($\frac{9.484}{2.64}$) compared to LB(Optimal) with $K = 8$ in Abilene but still achieves almost the same load-balancing performance as LB(Optimal) while LB-Successive Selection could further reduce computation times by a factor of 22.5X($\frac{9.484}{0.421}$) without losing load-balancing performance. As mentioned earlier, the computation time is only collected for the LP or MILP solver since they usually take much longer time compared to normal numerical computations. The results show that using different tie-breaker metrics (e.g., least-traffic, uniform, weighted) in LB-Successive Selection lead to very similar computation times.

Since the LB-Greedy algorithm exhaustively tests every individual monitor in each iteration and disables the least-impact one, its load-balancing performance is expected to be close to the optimal MILP solution. However it might not find the feasible solution when K is small since LB-Greedy only tests individual monitor instead of every possible combination. For example, LB-Greedy cannot find any feasible load-balancing solution when K=13 or 12 in GEANT since it disables the un-replaceable monitor[6] in the previous iterations as shown in Figure 3.7(b). The same drawback could be observed in LB-Successive Selection.

Note that LB-Greedy algorithm remains computationally costly, since it tests $O(\hat{M})$ monitors with $O(\hat{M})$ LP problems in each iteration. In GEANT with K=23, 22, 21, 20 and 19, the LB-Greedy approach even works slightly slower than the MILP solver, LB(Optimal). However for moderate size of limited monitors (e.g., K=14 to 19 out of M=23) in GEANT, LB-Greedy still can reduce computation time by a factor up to 65.8X($\frac{4417.08}{67.12}$) compared to LB(Optimal) when K=16 and achieves almost the same load-balancing performance. Furthermore, LB-Successive Selection could reduce computation times up to 800X($\frac{4417.08}{5.51}$) compared to LB(Optimal) without losing load-balancing performance in GEANT.

[6]In order to fulfill the measurement tasks imposed by the network operator, at least one capable monitor is needed in each OD-pair's routing path. If there is only one monitor left in its route, we denote it as un-replaceable monitor.

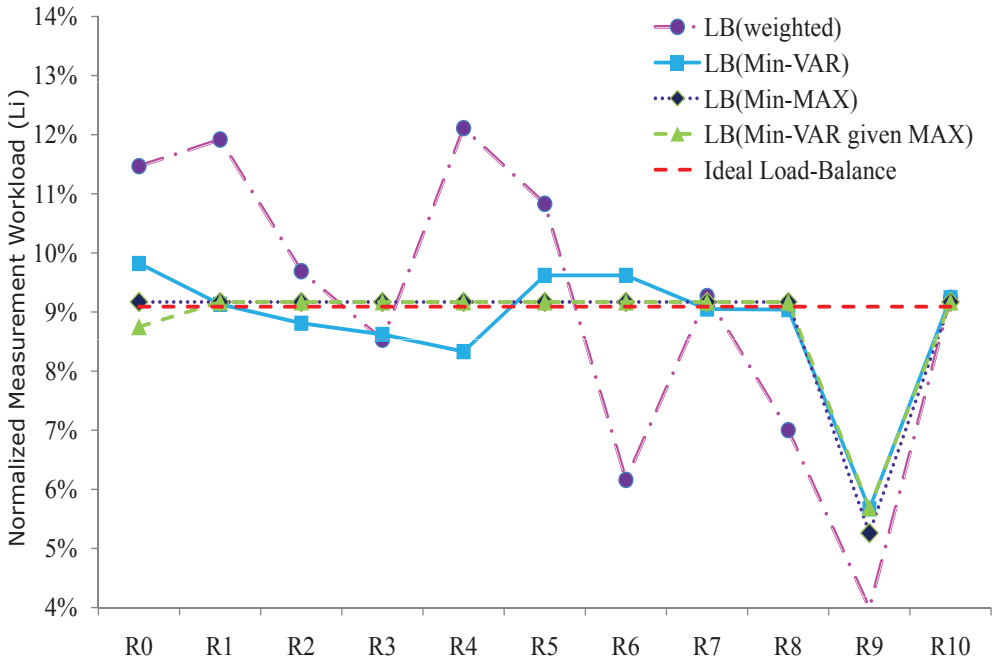


Figure 3.8: Measurement load distribution with multiple paths per OD-pair in Abilene.

3.6.4 Multiple Paths per OD-pair

Here we relax our second assumption to allow multi-path routing (e.g,. ECMP) for each OD-pair in Abilene network. In Figure 3.8, our proposed optimal solutions and heuristic approaches all have better load-balancing performance when applied in multi-path routing case compared to the single path routing. The rationale behind this is that with more overlaps in monitors/paths, LEISURE has more freedom (e.g., d_i^{xh} in Eq. (3.39)) to optimally load-balance the workloads across the participating monitors. The VAR(L_i) in multi-path case for LB(min-VAR), LB(min-MAX) and LB(min-VAR-given-MAX) is 0.0000917, 0.0000982 and 0.0000917, respectively while in the single path case is 0.000105, 0.000131 and 0.000105 in Figure 3.3.

3.6.5 Multiple Measurement Tasks

In this section, we examine our two-phase solution to the formulated joint optimization problem described in Section 3.5.3. Assume we have two measurement tasks with cost factor ratio C^1:C^2 and importance factor ratio I^1:I^2. Let θ reflect the

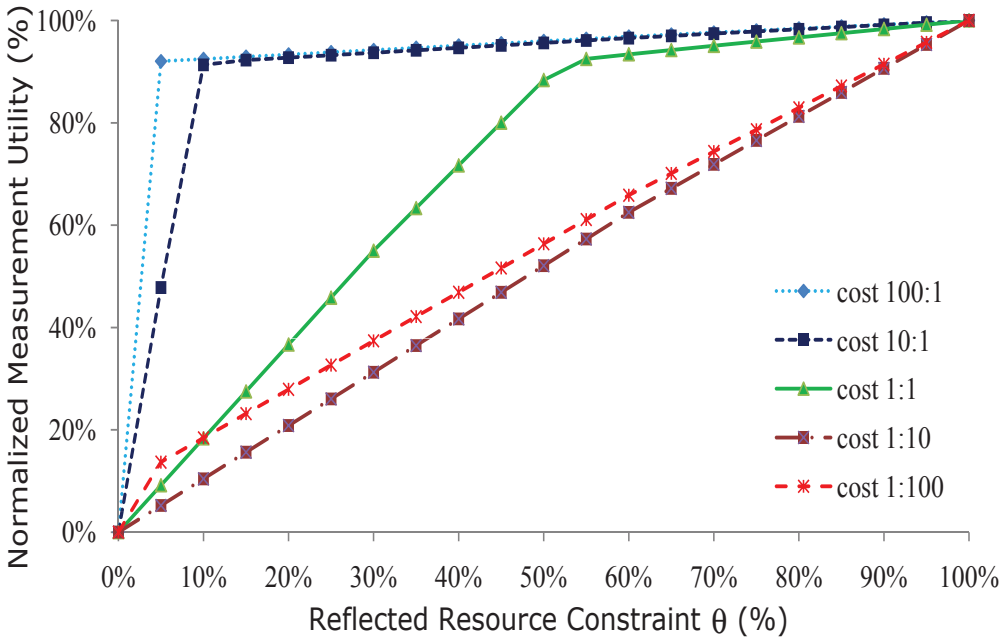


Figure 3.9: Two measurement tasks with different cost ratio and fixed importance ratio as 1:10.

identical resource constraint C_m for all routers V_i ($i = 1 \ldots M$) and represented as the fraction of L_{max}, the maximum workload routers can achieve derived from Eq. (3.40) to (3.43) where $0 \leq \theta \leq 1$. Figure 3.9 presents the result of our normalized measurement utility under different setup of C^1:C^2 (e.g., from 100:1 to 1:100) and fixed I^1:I^2=1:10 by changing resource constraint θ from 100% to 0%. Note that without resource constraint (i.e., $\theta = 100\%$), the normalized measurement utility LEISURE can achieve is always 1.0 (cover all measurement tasks).

As observed, if C^1:C^2 is directly proportional to I^1:I^2, the measurement utility decreases linearly when the resource constraint becomes severe (e.g., lower θ). On the other hand, if C^1:C^2 is inversely proportional to I^1:I^2, the measurement utility will not drop significantly until θ is extremely low (e.g., C^1:C^2=100:1 with I^1:I^2=1:10). This is because the optimal solution for Eq. (3.45) to (3.48) will let monitors always first fulfill the measurement request from the task with lower cost and higher importance. The other observation is that when C^1:C^2=1:1 and I^1:I^2=1:10, LEISURE can still remain 90% of

the measurement utility as in ideal case (i.e., without resource constraint) by using only half of the routers' resources (e.g., θ drops to 50%(=1/(1+1))). These results suggest that our framework can intelligently distribute measurement tasks for better load-balancing under resource constraints, while the overall measurement utility can still be preserved at a high level.

3.7 Conclusion

In this chapter, we proposed an optimization framework for load-balancing network-wide traffic measurements across coordinated monitors in the network while ensuring that the maximum traffic measurement utility of the network is achieved. This is an important problem because individual monitors are not capable of accomplishing the measurement tasks for all applications of interest due to its resource constraint, particularly resource-intensive measurement tasks such as those requiring deep packet inspection. Further, to uncover global network behavior, there is an inherent need to coordinate measurements among monitors distributed across the networks since the visibility of each monitor is only limited to the traffic that passes through it. Therefore, these distributed monitors can be coordinated for both coverage and optimized resource utilization. Based on our simulation measurement studies using the Abilene and GEANT networks, we found that our load-balancing optimization framework LEISURE can achieve up to 4.75X smaller maximum measurement workload and 70X smaller variance in workloads across all coordinated monitors. We also show that our proposed heuristic solutions could achieve almost the same load-balancing performance as the optimal solution, while reducing the computation times by a factor up to 22.5X in Abilene and 800X in GEANT under flexible monitor deployment scenario. The distributed LEISURE algorithm for load balancing problem is deferred as our future work.

Chapter 3, in part, is a reprint of the material as it appears in the following publications:

- Chia-Wei Chang, Guanyao Huang, Bill Lin and Chen-Nee Chuah, "LEISURE: A Framework for Load-Balanced Network-Wide Traffic Measurements",

*ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS),*Brooklyn, NY, October 3-4, 2011.

Chapter 3, in full, has been submitted for publication of material as it may appear in IEEE Transactions on Network and Service Management (TNSM), Chia-Wei Chang, Guanyao Huang, Bill Lin and Chen-Nee Chuah, “A Joint Optimization Approach for Load-Balanced Network-Wide Traffic Measurements and Monitor Placement”. The dissertation author was the primary investigator and author of the papers.

Chapter 4

Distributed Measurement-Aware Routing: Striking a Balance between Measurement and Traffic Engineering

4.1 Introduction

Achieving accurate and efficient network-wide traffic measurement is often plagued with multi-faceted challenges. While packet and flow sampling mechanisms are widely deployed (e.g, NetFlow) [29], detailed packet capture and analysis (e.g., deep packet inspection [25]) is computationally expensive. Hence, typically only a subset of nodes are equipped with such high-fidelity monitoring capabilities. To reap the maximum measurement benefits without incurring huge deployment costs, these high-fidelity monitors need to be configured properly and strategically placed across the network. Most previous work on the latter domain focused on deriving the optimal monitor placement that maximizes the monitoring utility for a given routing and traffic profile. They are typically intended for longer time-scales and assume a priori knowledge about the traffic characteristics. However, both traffic characteristics and measurement objectives can dynamically change over time, rendering a carefully designed placement of monitors sub-optimal. To address these limitations, a measurement-aware routing framework, MeasuRouting, was recently proposed to assist traffic measurement [63].

It introduces routing as another degree of freedom and intelligently routes traffic sub-populations over pre-deployed monitors to maximize the traffic measurement gain. However, MeasuRouting requires the existence of some centralized controller and offline analysis to find the optimal routing strategies for every router, which is unrealistic in production IP networks. It can therefore only be interpreted as the best-case bounds for routing-assisted measurement.

In order to gain comparable measurement improvement as the centralized approach, a de-centralized (distributed) measurement-aware routing solution faces two main challenges. First, individual nodes (being selfish) tend to compete unknowingly for the limited monitoring resources. A natural solution to such problem is to design a game-theory based distributed algorithm to achieve some equilibrium point. The solution should not only guarantee the existence of an equilibrium point, but also provide fast convergence. While similar in spirit, existing work on selfish routing tends to minimize a single TE cost function of the network, e.g., path delay or average link utilization [28, 32, 46]. Distributed measurement-aware routing, on the other hand, needs to maximize measurement gains while adhering to traffic engineering constraints, which cannot be achieved easily with existing approaches. Second, individual nodes need to make local routing decisions based on local (partial) view of the network conditions, including the "possible" measurement gain and link congestions. A light-weight protocol is needed to distribute accurate measurement and load information across the network.

In this chapter, we present *Distributed MeasuRouting (DisMR)*, a new traffic engineering protocol that attempts to optimally utilize existing monitor locations for maximizing the traffic measurement gain while distributing the traffic load evenly across the network. DisMR takes advantage of alternative paths in a network (e.g., ECMP multipaths). It maximizes the traffic measurement gain by adjusting the traffic split ratios among these paths to the same destination. It actually operates on top of an existing ECMP infrastructure. DisMR is derived from a game-theoretic re-routing policy that captures the dynamic decision-making process and interactions among distributed routers. In our model, we design a cost function on each link that reflects both the measurement capability and TE constraint, i.e., links with larger measurement

resources have a smaller cost but links with a larger TE score (e.g., link utilization) have a larger cost. The cost function is designed such that flows are attracted to links with better measurement capabilities while avoiding TE violations. Routers compete with each other in a game-theoretic manner in order to minimize their own costs for the downstream paths. In DisMR, every router periodically gathers/propagates sub-path cost information for upstream routers. Based on this information, each router makes local decisions on how to adjust routing split ratios for each destination traffic to the next-hop routers among these multiple equal-cost paths. Our routing policy guarantees not only a provable Nash equilibrium but also a fast convergence without significant oscillations. Meanwhile, the measurement gain of the network at the equilibrium state is close to the maximum achievable gain calculated using offline, centralized MeasuRouting. We outline our contributions as follows:

- We de-centralize MeasuRouting in a game-theoretic setting and propose a novel cost function that balances the potentially contradicting measurement and traffic engineering objectives. The cost function is designed to encourage flows to be routed through monitors with abundant resources while avoiding TE violation. We prove the existence of Nash equilibrium and derive bounds on the price of anarchy (POA) for the game.

- We design a new traffic engineering protocol, *Distributed MeasuRouting (DisMR)*, based on the routing game. DisMR converges fast to equilibrium point and achieves comparable measurement gain with centralized MeasuRouting in static traffic scenario.

- We evaluate DisMR via simulations using both synthetic and real traces/topologies from Abilene [1], AS6461 [69], and GEANT [3]). The simulation results show fast convergence (as expected from the theoretical results), improved measurement gains (e.g., 12 % higher) and much lower TE-violations (e.g., up to 100X smaller) compared to static, centralized MeasuRouting in dynamic traffic scenario.

The rest of this chapter is organized as follows: We first prove the existence of equilibrium on the new cost function in Section 4.2. We next study the rerouting

policies in a "dynamic round-based" variant of equilibrium in Section 4.3. We present practical *Distributed MeasuRouting* algorithm in Section 4.3.1 and prove it to be stable and converge quickly in a game-theoretic model under realistic conditions. Section 4.4 presents detailed performance evaluation of our proposed algorithm. Section 4.5 discusses related work and Section 4.6 concludes this chapter.

4.2 Adaptive traffic measurement problem

In this section, we formulate the Distributed MeasuRouting problem in a game-theoretic setting. It strikes the balance between measurement and TE constraints by introducing two novel definitions: Ψ (effective non-sampling rate) and ζ (link penalty function). We present theoretical results regarding the static convergence of the game. Note that our work differs fundamentally from Beckmann's work in that our introduced link cost function is a novel combination of link measurement ability and TE constraint. Moreover, the path cost is defined as the product of link costs, which makes the proofs of existence of Nash Equilibrium and POA different from [11]. The dynamic behavior of this game and its distributed implementation are presented in next section.

We consider a measurement objective of maximizing G (*sampling resolution function*), which characterizes the overall measurement utility of the whole network as MeasuRouting used [63]. In contrast to MeasuRouting, we assume independent uniform sampling across every link, the de-facto implemented method in current Internet. Let S_a be the given fixed sampling rate at every arc $a \in \mathcal{A}$. The total effective sampling rate of a path $P \in \mathcal{P}$ with respect to flow set, $[f] = \{f_P, P \in \mathcal{P}\}$ is defined as: $S_P(f_P) = 1 - \prod_{a \in P} (1 - S_a)$. Therefore $G(f) = \sum_{P \in \mathcal{P}} S_P(f_P) \cdot f_P$. We define Ψ_a to be the effective non-sampling rate at arc $a \in \mathcal{A}$: $\Psi_a = 1 - S_a$. The total non-sampling rate of a path $P \in \mathcal{P}$ with respect to f_P is then the product of the non-sampling rate of the arcs on that path: $\Psi_P(f_P) = \prod_{a \in P} \Psi_a(f_P), P \in \mathcal{P}$. Therefore the total non-sampled amount is defined as $C(f) = \sum_{P \in \mathcal{P}} \Psi_P(f) \cdot f_P$. Given fixed traffic demand, maximizing $G(f)$ could be equivalent to minimize the cost function $C(f)$.

Our goal is to let the flow sets at each end point route their traffic selfishly to better learn a Nash equilibrium of non-sampling rate while adhering to traffic

engineering constraints. However, in a distributed environment, flow sets will all choose the best paths with minimum $\Psi_P(f)$ and may overload some specific arcs. This is because Ψ_a at every arc $a \in \mathcal{A}$ is constant (e.g., sampling rates do not adapt to the traffic amount). In order to reflect TE constraints, we add penalty function $\zeta(f)$ to Ψ_a, i.e., $\Psi_a(f) = \Psi_a + \zeta(f)$ for each arc $a \in \mathcal{A}$. We design the $\zeta(f)$ such that its value increases sharply when the traffic amount is above the TE-constraint (e.g., maximum link utilization), otherwise it will stay at zero. Therefore, $\Psi_a(f)$ becomes a function of traffic for every arc $a \in \mathcal{A}$ (i.e., a non-decreasing and continuous function).

Suppose every flow set tends to minimize its own cost, $C(f_P) = \Psi_P(f_P) \cdot f_P$, we prove the existence of static Nash equilibrium for this game in Section 4.2.1 and the optimal flow in Section 4.2.2. The details about how to design the penalty function are discussed in Section 4.2.3.

4.2.1 The Existence of Nash Equilibrium

We consider a model for selfish routing where each of an infinite population of agents wants to send an infinitesimal amount of traffic (flows) through a network $G = (V, \mathcal{A})$ with vertex set V, arc set $\mathcal{A}$, and k source-to-destination vertex pairs, $\{s_i, t_i\}, i \in [k] = \{1, \ldots, k\}$ with flow demand r_i. Each agent belongs to one of the $\{s_i, t_i\}, i \in [k]$. Let $\mathcal{P}_i$ denotes the set of multiple equal-cost routing paths from s_i to t_i in G and $\mathcal{P} = \bigcup_i \mathcal{P}_i$, the set of all possible routing paths. The flow set $f_P, P \in \mathcal{P}$ is feasible if for all $i \in [k]$, $\sum_{P \in \mathcal{P}_i} f_P = r_i$. For a given flow set $f_P, P \in \mathcal{P}$, we define the aggregated flows on arc $a \in \mathcal{A}$ as $f_a = \sum_{P \in \mathcal{P}: a \in P} f_P$. The non-sampling rate of a path $P \in \mathcal{P}$ is $\Psi_P(f) = \prod_{a \in P} \Psi_a(f)$ where $\Psi_a(f) = \Psi_a + \zeta(f)$ for each arc $a \in \mathcal{A}$. We are interested in flow assignments that are stable in the sense that no agent can improve their $\Psi_P(f)$ by changing their paths selfishly.

Definition 1. *A feasible flow set $f_P, P \in \mathcal{P}$ is at a Wardrop (Nash) equilibrium if for each $i \in [k]$ and every path $P, R \in \mathcal{P}_i$ with $f_P > 0$, it holds that $\Psi_P(f) \leq \Psi_R(f)$.*

To prove that the Nash flows always exist in our non-sampling rate case and the achieved cost is unique, we use the Karush-Kuhn-Tucker optimality conditions as in the studies by Beckmann et al. [11] and Dafermos et al. [24]. Let $Q_a(x) = \ln(\Psi_a(x))$ for

every arc $a \in \mathcal{A}$ (i.e., also non-decreasing and continuous). Similar to [11] and [24], we construct a convex program (CP) as following with continuously differentiable and convex functions $(h_a)_{a\in\mathcal{A}}$, which is defined as $h_a(f_a) = \int_0^{f_a} Q_a(x)dx$:

$$\texttt{Minimize} \sum_{a\in\mathcal{A}} h_a(f_a) \tag{4.1}$$

$$s.t. \sum_{P\in\mathcal{P}_i} f_P = r_i \qquad \forall i \in [k] \tag{4.2}$$

$$f_a = \sum_{P\in\mathcal{P}:a\in P} f_P \qquad \forall a \in \mathcal{A} \tag{4.3}$$

$$f_P \geq 0 \qquad \forall P \in \mathcal{P} \tag{4.4}$$

$$h_a(f_a) = \int_0^{f_a} Q_a(x)dx \tag{4.5}$$

Based on the Karush-Kuhn-Tucker optimality conditions, a feasible flow set $f_P, P \in \mathcal{P}$ is an optimal solution for this convex program if and only if

$$\forall i \in [k], \forall P, R \in \mathcal{P}_i, f_P > 0 \tag{4.6}$$

$$h'_P(f) = \sum_{a\in\mathcal{P}} h'_a(f_a) \leq \sum_{a\in\mathcal{R}} h'_a(f_a) = h'_R(f), \tag{4.7}$$

where $h'_a(x)$ refers to the first derivative of $h_a(x)$. Therefore

$$h'_P(f) = \sum_{a\in\mathcal{P}} h'_a(f_a) = \sum_{a\in\mathcal{P}} Q_a(f_a) = \sum_{a\in P} \ln(\Psi_a(f_a)) \tag{4.8}$$

$$= \ln(\prod_{a\in P} \Psi_a(f_a)) = \ln(\Psi_P(f)) \tag{4.9}$$

$$\leq \sum_{a\in\mathcal{R}} h'_a(f_a) = \sum_{a\in\mathcal{R}} Q_a(f_a) = \sum_{a\in R} \ln(\Psi_a(f_a)) \tag{4.10}$$

$$= \ln(\prod_{a\in R} \Psi_a(f_a)) = \ln(\Psi_R(f)) \tag{4.11}$$

It means $\ln(\Psi_P(f_a)) \leq \ln(\Psi_R(f_a))$, which implies $\Psi_P(f_a) \leq \Psi_R(f_a)$ for $\forall i \in [k], \forall P, R \in \mathcal{P}_i, f_P > 0$. The optimality condition of the convex problem coincides with the condition of the Nash equilibrium.

4.2.2 The Existence of Optimal Flow

An optimal flow is defined as a feasible flow set $[f] = \{f_P, P \in \mathcal{P}\}$ that minimizes the total system cost $C(f)$ instead of letting all agents selfishly minimize their own cost $C(f_P)$[1]. Recall that the total cost with respect to f is defined as:

$$C(f) = \sum_{P\in\mathcal{P}} \Psi_P(f) f_P = \sum_{P\in\mathcal{P}} (\prod_{a\in P} \Psi_a(f_a)) f_P \tag{4.12}$$

$$= \sum_{P\in\mathcal{P}} (\prod_{a\in P} \Psi_a(\sum_{P\in\mathcal{P}:a\in P} f_P)) f_P \tag{4.13}$$

If we replace the objective function in Equation (4.1) to minimize $C(f)$ instead, the optimal solution, $f_P, P \in \mathcal{P}$, to this new problem becomes the optimal flow. In our case, $C(f)$ is the summation of product of numerous decision variables as Equation (4.13), which can not be easily solved.

By applying *Inequality of Arithmetic and Geometric Means*, we give the lower bounds of $C(f)$ as follows. We first take natural log of the total cost, $C(f)$, and $\ln(C(f))$ could be simplified as following by using $\ln(a_1 + a_2 + \ldots + a_N) \geq \ln(N) + \frac{\ln(a_1)+\ln(a_2)+\ldots+\ln(a_N)}{N}$ since $\frac{a_1+a_2+\ldots+a_N}{N} \geq \sqrt[N]{a_1 \cdot a_2 \cdot \ldots \cdot a_N}$ if $a_1, a_2, \ldots, a_N \geq 0$ and $N \in \mathcal{N}$.

$$\begin{aligned}
\ln(C(f)) &= \ln\left(\sum_{P\in\mathcal{P}} \Psi_P(f) f_P\right) = \ln\left(\sum_{P\in\mathcal{P}} \prod_{a\in P} \Psi_a(f_a) \cdot f_P\right) \\
&\geq \ln(N) + \frac{1}{N} \cdot \left(\sum_{P\in\mathcal{P}} \ln(\prod_{a\in P} \Psi_a(f_a) \cdot f_P)\right) \\
&= \ln(N) + \frac{1}{N} \cdot \left(\sum_{P\in\mathcal{P}} (\sum_{a\in P} \ln(\Psi_a(f_a)) + \ln(f_P))\right) \\
&= \ln(N) + \frac{1}{N} \cdot \left(\sum_{P\in\mathcal{P}} \sum_{a\in P} \ln(\Psi_a(f_a)) \cdot 1 + \sum_{P\in\mathcal{P}} \ln(f_P)\right) \\
&= \ln(N) + \frac{1}{N} \cdot \left(\sum_{a\in\mathcal{A}} n_a \cdot \ln(\Psi_a(f_a)) + \sum_{P\in\mathcal{P}} \ln(f_P)\right) \\
&= \ln(N) + \frac{1}{N} \cdot \left(\sum_{a\in\mathcal{A}} \ln(\Psi_a(f_a)) \cdot n_a + \sum_{P\in\mathcal{P}} \ln(f_P)\right) \\
&= \ln(N) + \frac{1}{N} \cdot (C^*(f))
\end{aligned}$$

[1]The ratio between the total cost at Nash equilibrium and optimal flow is usually called POA (Price of Anarchy) in [27]

where N is the number of path $P \in \mathcal{P}$ and $n_a = \sum_{P \in \mathcal{P}, f_P > 0: a \in P} 1$, is the number of *effective* paths passing through arc a with traffic $f_P > 0, P \in \mathcal{P}$. We use boolean variable, u_P, to indicate whether the routing policy uses path P, i.e., $u_P = 1$ if using path $P \in \mathcal{P}$ ($f_P > 0$), otherwise $u_P = 0$ if ($f_P = 0$). Therefore, $n_a = \sum_{P \in \mathcal{P}: a \in P} u_P, \forall a \in \mathcal{A}$ and the constraints Equation (4.2)-(4.3) become:

$$\sum_{P \in \mathcal{P}_i} f_P \cdot u_P = r_i \qquad \forall i \in [k] \tag{4.14}$$

$$f_a = \sum_{P \in \mathcal{P}: a \in P} f_P \cdot u_P \qquad \forall a \in \mathcal{A} \tag{4.15}$$

Since N is constant given by the network topology, the lower bound objective function could be equivalent to minimize $C^*(f)$ where $C^*(f) = \sum_{a \in \mathcal{A}} \ln(\Psi_a(f_a)) \cdot n_a + \sum_{P \in \mathcal{P}} \ln(f_P)$. In order to avoid quadratic programming, we introduce $z_P = f_P \cdot u_P$ to decouple $f_P \cdot u_P$ in Equation (4.16)-(4.17). It is easy to see their equivalence. When $u_P = 0$, $z_P = 0$ from Equation (4.16); and when $u_P = 1$,$z_P = f_P$ from Equation (4.17).

$$0 \leq z_P \leq u_P \tag{4.16}$$

$$f_P + u_P - 1 \leq z_P \leq f_P \tag{4.17}$$

After we substitute z_P to Equation (4.14)-(4.15), the convex program could be simplified as:

$$\texttt{Minimize } C^*(f) \tag{4.18}$$

$$s.t. \sum_{P \in \mathcal{P}_i} z_P = r_i \qquad \forall i \in [k] \tag{4.19}$$

$$f_a = \sum_{P \in \mathcal{P}: a \in P} z_P \qquad \forall a \in \mathcal{A} \tag{4.20}$$

$$f_P > 0 \qquad \forall P \in \mathcal{P} \tag{4.21}$$

$$n_a = \sum_{P \in \mathcal{P}: a \in P} u_P \qquad \forall a \in \mathcal{A} \tag{4.22}$$

$$u_P \in \{0, 1\} \qquad \forall P \in \mathcal{P} \tag{4.23}$$

$$0 \leq z_P \leq u_P \qquad \forall P \in \mathcal{P} \tag{4.24}$$

$$f_P + u_P - 1 \leq z_P \leq f_P \qquad \forall P \in \mathcal{P} \tag{4.25}$$

By solving the new convex optimization problem, we achieve the lower bound for the original optimal flow problem and hence derive the POA as follows. Since $\ln(C(f)) \geq$

$\ln(N) + \frac{C^*(f)}{N}$, the lower bound of $C(f)$ with respect to the optimal flow, f^* could be expressed as $C(f^*) \geq N \cdot e^{\frac{C^*(f^*)}{N}}$. Let $(G, \mathcal{A}, \Psi)$ be an instance of the selfish routing game and assume f and f^* be a Nash flow and an Optimal flow, respectively. The upper bound of the price of anarchy in our case is:

$$\rho(G, r, \Psi) = \frac{C(f)}{C(f^*)} \tag{4.26}$$

$$\leq \frac{C(f)}{N \cdot e^{\frac{C^*(f^*)}{N}}} \tag{4.27}$$

4.2.3 Design of Penalty Functions

In the routing game, after the current link capacity U_a exceeds U_{max}, we add a sharp penalty to the metric $\Psi_a(f)$ such that selfish agents are aware of the TE constraints. The more U_a exceeds U_{max}, the larger the penalty $\Psi_a(f)$ will be. $U_a = \frac{f}{C_a}$, where C_a is the link capacity and f is the current traffic on link a. Here we use *additive* operator to embed penalty function $\zeta(f)$ into $\Psi_a(f)$, i.e., $\Psi_a(f) = (1 - S_a) + \zeta(f)$. We keep $\zeta(f) = 0$ if $U_a < U_{max}$ and make $\zeta(f)$ increase sharply if $U_a \geq U_{max}$ as follows:

$$\zeta(f) = \begin{cases} 0, & \text{if } U_a < U_{max}; \\ (U_a - U_{max}) \cdot m_\zeta, & \text{if } U_a \geq U_{max}; \end{cases}$$

and therefore

$$\Psi_a(f) = \begin{cases} (1 - S_a) + 0, & \text{if } \frac{f}{C} < U_{max}; \\ (1 - S_a) + (\frac{f}{C} - U_{max}) \cdot m_\zeta, & \text{if } \frac{f}{C} \geq U_{max} \end{cases}$$

where m_ζ controls the sharpness of the penalty. Usually with a larger m_ζ, it will have fewer TE-violations in the equilibrium state but with longer convergence time. We find $m_\zeta = 10^6$ provides a good trade-off between those two effects described above. Section 4.4.2 compares the link utilization, convergence speed, and the effective measuring gains with different m_ζ values.

4.3 round-based equilibrium in IP network

Up to this point, our traffic model is based on the assumption that agents at end hosts have full control over their traffic and they can access the current TE cost value of all paths. Obviously, none of these is true in the real-world IP networks. In this section, we study our Nash equilibrium model that both considers effective non-sampling rate and TE-violation penalty in a dynamic/distributed, round-based variant. Suppose agents at end hosts are activated every T_s seconds and are allowed to change their routes simultaneously. Since they all intend to migrate traffic to a path with minimal cost value, such global migration behavior will result in greatly increased congestion on the optimal path (from measurement's perspective) and lead to oscillations. Fischer et al. proposed the so-called (α-β)-exploration-replication policy in [33] to avoid traffic migration oscillation by using adaptive path-sampling methods. Although [33] is designed for the cost model defined for latency, we apply it to our newly defined non-sampling rate cost model.

In real IP networks, agents at the end points do not have the routing control over the paths; it is the routers that are responsible for choosing the paths. A router can only determine the next-hop nodes to the destination and decide their traffic split ratios. In this section, we provide an overview of our distributed routing policy: how routers propagate/gather cost information and determine the proper split ratios. The detailed algorithms are deferred to the next subsection.

Consider an intermediate router R on a path $S \rightarrow R \rightarrow D$ from S to D and assume there are several multiple equal-cost paths between R to D. Applying our Wardrop rerouting policy based on minimizing path non-sampling rate, router R aims at distributing the traffic from S to D evenly among these paths, however, R only knows the set of possible next hops of these multiple paths (denoted by $N(R, D)$) to each destination D. Therefore router R needs to maintain a set of dynamically changeable weights $w(R, D, V_i)$ for all possible next-hop routers $V_i \in N(R, D)$ to every destination D and $\sum_{V_i \in N(R,D)} w(R, D, V_i) = 1$. $w(R, D, V_i)$ can be treated as the fraction of traffic routed from R to D via V_i.

In this model, the routing decision made at intermediate router R only affects the performance of the paths from R to D while the performance between S to R is fixed and unaffected. Note that R can not explore the traffic information for downstream

routers between V_i and D. It needs to utilize the aggregated traffic information from next-hop routers V_i instead. This information exchange resembles a distance vector routing protocol. The decision for R to route traffic via V_i depends on the *expected non-sampling rate* $\Psi(R, D, V_i)$ for the path $R \rightarrow V_i \dashrightarrow D$. Each round, every router V_i keeps R informed about the *expected non-sampling rate* of its paths $V_i \dashrightarrow D$. We denote the set of multiple equal-cost paths between node V_i and D as $\mathcal{P}(V_i, D)$ and N_P is the number of cascaded intermediate routers on each path $P \in \mathcal{P}(V_i, D)$. The *expected non-sampling rate* $\Psi(R, D, V_i)$ via V_i can be expressed as

$$\Psi(R, D, V_i) = \Psi(R, V_i) \cdot (\sum_{P \in \mathcal{P}(V_i, D)} w_P \cdot \Psi_P) \tag{4.28}$$

$$w_P = \prod_{j=1}^{N_P} w(V_j, D, V_{j+1}), \forall P \in \mathcal{P}(V_i, D) \tag{4.29}$$

,where $\Psi(R, V_i)$ is the measured non-sampling rate on link $R \rightarrow V_i$. Ψ_P is the non-sampling rate on path P, and w_P is the traffic fraction on path P. Let $A(V_i, D)$ indicate the information that router V_i feedback to R:

$$A(V_i, D) = \sum_{P \in \mathcal{P}(V_i, D)} w_P \cdot \Psi_P \tag{4.30}$$

Assume there are several multiple equal-cost paths between V_i to D, and $N(V_i, D)$ denote the set of possible next-hop routers to destination D, we can obtain

$$\begin{aligned} A(V_i, D) &= \sum_{U_j \in N(V_i, D)} w(V_i, D, U_j) \sum_{P \in \mathcal{P}(U_j, D)} w_P \cdot (\Psi(V_i, U_j) \cdot \Psi_P) \\ &= \sum_{U_j \in N(V_i, D)} w(V_i, D, U_j) \sum_{P \in \mathcal{P}(U_j, D)} w_P \cdot \Psi_P \cdot \Psi(V_i, U_j) \\ &= \sum_{U_j \in N(V_i, D)} w(V_i, D, U_j) (\sum_{P \in \mathcal{P}(U_j, D)} w_P \cdot \Psi_P) \cdot \Psi(V_i, U_j) \\ &= \sum_{U_j \in N(V_i, D)} w(V_i, D, U_j) \cdot \Psi(V_i, D, U_j) \end{aligned} \tag{4.31}$$

In summary, the value of $A(V_i, D)$ is computed at router V_i each round and sent back to R based on the previous known information of $\Psi(V_i, D, U_j)$. $A(V_i, D)$ can be treated as the condensed information of *expected non-sampling rate* beyond V_i. The router R can then update $\Psi(R, D, V_i)$, the *expected non-sampling rate* to destination D via V_i, and

compute $w(R, D, V_i)$ to adjust its split ratios. Here we assume synchronized routing-updates of these link/path costs. The impacts of asynchronous update issue could be solved similarly in [28] where we defer as our future work.

4.3.1 Distributed MeasuRouting Algorithm

In this section, we present our adaptive algorithm, *Distributed MeasuRouting (DisMR)*, which runs on each individual routers to make routing decisions on how to adjust routing split ratios for each destination traffic. In order to do this, each router first needs to measure the non-sampling rate $\Psi(R, V_i)$ for each link to next-hop routers V_i and exchanges information with other routers by using *Distributed* Ψ*-Propagation Algorithm*. After receiving $A(V_i, D)$, *the expected average non-sampling rate* of the path to every destination D via V_i from next-hop routers, each router can compute $\Psi(R, D, V_i)$ locally and use this information to conduct the *Adaptive Weight Calculations*. In summary, each router R needs to maintain the following sets of information for all possible next-hop routers $V_i \in N(R, D)$ to every destination D:

1. $\Psi(R, V_i)$: the non-sampling rate value that also includes the penalty value to reflect the current link utilization on link $R \rightarrow V_i$.

2. $A(V_i, D)$: the expected average non-sampling rate value to destination D via V_i ($V_i \dashrightarrow D$) which is received periodically from neighbor router V_i.

3. $w(R, D, V_i)$: current dynamically changeable weights for traffic routed from current router R to destination D via V_i.

Algorithm 6 describes the distributed Ψ-metric propagation procedure of DisMR in details. Every T_s seconds, the set of $\Psi(R, D, V_i)$ values are updated at each router by using the information of current $\Psi(R, V_i)$ and previous $A(V_i, D)$ from neighbors (Line 7). Subsequently, the new $A(R, D)$ values are re-calculated by using the current weights $w(R, D, V_i)$ and broadcast to all of the neighbor routers (Line 9-10). Meanwhile each router will execute the Adaptive Weight Calculation procedure to reassign the weights $w(R, D, V_i)$ for all possible next-hop routers $V_i \in N(R, D)$ to every destination D by using updated information of $\Psi(R, D, V_i)$ (Line 12).

Algorithm 6 Distributed Ψ-Propagation Algorithm

1: assume current node is R
2: **while** every T_s secs **do**
3: initialize new update message $M(T_s)$
4: **for** each destination D in routing table **do**
5: **for** every next-hop nodes $V_i \in N(R, D)$ **do**
6: measure $\Psi(R, V_i)$
7: $\Psi(R, D, V_i) = \Psi(R, V_i) \cdot A(V_i, D)$
8: **end for**
9: $A(R, D) = \sum_{V_i \in N(R,D)} w(R, D, V_i) \cdot \Psi(R, D, V_i)$
10: Append $A(R, D)$ in $M(T_s)$
11: **end for**
12: Execute one of the **Adaptive-Weights calculations**
13: Send $M(T_s)$ to all neighbor nodes
14: After receiving $M(T_s)$ from neighbor node U_i
15: **for** each $A(U_i, D)$ in $M(T_s)$ **do**
16: **if** $U_i \in N(R, D)$ **then**
17: Update $A(U_i, D)$ from $M(T_s)$
18: **end if**
19: **end for**
20: **end while**

Algorithm 7 presents the Adaptive Weight Calculation procedure of DisMR. For every pair of next-hop routers (e.g., say V_1, V_2), it first compares their cost metric $\Psi(R, D, V_i), i = 1, 2$ and conducts the migration procedure if the difference of their cost values is more than the *migration threshold* ($\epsilon \times m_\zeta$) (Line 7). Otherwise, DisMR will not change the weights of V_1 and V_2.

Subsequently, it computes the migration probability (Line 7-9) and the adaptive migration amount (Line 10-14) according to the (α-β)-exploration-replication policy [33]. For every pair of next-hop nodes in each round (Line 3), we denote V_1 to be the node with larger cost value, $\Psi(.)$ and V_2 to be the alternate node. From statistic

Algorithm 7 Adaptive Weight Calculation

1: after $\Psi(R, D, V_i)$ information is updated
2: **for** each destination D in routing table **do**
3: **for** every next-hop node $V_i \in N(R, D)$ **do**
4: $w_{new}(R, D, V_i) = w(R, D, V_i)$
5: **end for**
6: **for** every pair of next-hop nodes $V_1, V_2 \in N(R, D)$ **do**
7: **if** $\Psi(R, D, V_1) > \Psi(R, D, V_2) + \epsilon \times m_\zeta$ **then**
8: Calculate $P_M = \frac{\Psi(R,D,V_1)-\Psi(R,D,V_2)}{\Psi(R,D,V_1)+\alpha}$
9: **if** with probability P_M **then**
10: **if** $w(R, D, V_2) \neq 0$ **then**
11: $\Delta = (1 - \beta) \cdot w(R, D, V_2) \cdot \Delta_{fix}$
12: **else**
13: $\Delta = \frac{\beta}{N(R,D)} \cdot \Delta_{fix}$
14: **end if**
15: $w_{new}(R, D, V_1) = w(R, D, V_1) - \Delta$
16: $w_{new}(R, D, V_2) = w(R, D, V_2) + \Delta$
17: **end if**
18: **end if**
19: **end for**
20: Use $w_{new}(R, D, V_i)$ to distribute the traffic
21: **end for**

point of view, the adaptive migration amount Δ should be calculated depending on node V_2. If V_2 is already used (e.g., $w(R, D, V_2) \neq 0$), then $\Delta = (1 - \beta) \cdot w(R, D, V_2) \cdot \Delta_{fix}$ from *proportional sampling* perspective. If V_2 is unused (e.g., $w(R, D, V_2) = 0$), then $\Delta = \frac{\beta}{N(R,D)} \cdot \Delta_{fix}$ from *uniform sampling* perspective where Δ_{fix} is the unit of weight shifted in one round. The migration probability is decided as $P_M = \frac{\Psi(R,D,V_1)-\Psi(R,D,V_2)}{\Psi(R,D,V_1)+\alpha}$ based on [33] in order to avoid oscillations from global synchronized migrations (Line 8). This adaptive migration policy ensures that smaller non-sampling rate gains, $\Delta_\Psi = \Psi_P - \Psi_Q$, only cause a smaller migration possibility and avoid oscillation. The implementation of distributing traffic according to $W(R, D, V_i)$ for each router can use

the hashing methods described in [28, 32, 46]. If $W(R, D, V_i)$ are constant, there is no packet reordering occurred. However once $W(R, D, V_i)$ are shifted, a fraction of the traffic needs to be rerouted and probably causes packet reordering. The solution is to make the time interval when $W(R, D, V_i)$ shifts occur not smaller than the time TCP needs to recover from packet losses in [32]. We summarize the static parameters used in DisMR as follows.

- *Update interval* T_s: it controls how often the participated routers update their traffic split ratios.
- *Migration threshold* $\epsilon \times m_\zeta$: it controls the granularity of equilibrium DisMR wants to achieve where m_ζ is the severeness of the penalty and ϵ is the inaccurate-rate we can tolerate.
- *Virtual non-sampling rate offset* α and *Exploration-replication factor* β: They are used to control (α-β)-exploration-replication policy to avoid traffic migration oscillation (details are in [33]).
- *Migration rate* Δ_{fix}: the unit of weight shifted in one round. It controls the convergence speed of DisMR (details are discussed in Section 4.4.2).

4.4 Performance Evaluation

In this section, we evaluate DisMR using both synthetic and real traces. We consider various topologies ranging from a simple 4 node topology, Abilene [1], GEANT [3] (with 23 nodes and 74 links) to AS6461 topology obtained using RocketFuel (with 19 nodes and 68 links) [69]. In each set of topology, we first calculate multiple paths for every OD (origin-destination) pair nodes to simulate the (ECMP)-like algorithm in practical scenarios, and run DisMR on those multiple paths. Our simulations have three goals: (1) determine good parameters for the algorithm to quickly reach equilibrium state without significant oscillations; (2) show that the measurement gain of the network at equilibrium state is close to the offline maximum achievable gain calculated by static centralized MeasuRouting; (3) show that it indeed

improves measurement gain in dynamic traffic scenario compared to static centralized MeasuRouting.

4.4.1 Traces and Performance Metrics

We use the following four topologies in our experiments and assume there are only one class of traffic in this chapter.

- Simple 4-node topology: As shown in Figure 4.1, all links have 10Gbps link capacity and the TE-constraint (maximum link-utilization) is U_{max}=0.9. The traffic demand is 15Gbps from SF$\rightarrow$NY with two multiple paths. Two links are equipped with monitors as sampling rate: $P_{SF\rightarrow A} = 0.5$ and $P_{SF\rightarrow B} = 0.7$, respectively.

- Abilene: This is a public academic network in the U.S. with 11 nodes interconnected by 28 OC192 (10 Gbps) links. The traces we use were collected from April 22-26, 2004 [1].

- AS6461: This is a RocketFuel [69] topology with 19 nodes and 68 links. We generate aggregate traffic demands for each OD pair using the Gravity Model [55].

- GEANT: GEANT connects a variety of European research and education networks. Our experiments are based on the December 2004 snapshot available at [3], which consists of 23 nodes and 74 links varied from 155 Mbps to 10 Gbps. The traces we use were collected from April 11-15, 2004.

In our experiments, we are interested in the following four performance metrics[2]:

1. Convergence Time: It is defined as the number of iterations the algorithm takes to reach equilibrium state. Note that the actual convergence time should be the product of T_s and the number of iterations.

2. Measurement Gain: It is defined as the sum of sampling utilities ($S_P(f)$) of flows weighted by the flow sizes (f_P) along the paths, $G = \sum_{P\in\mathcal{P}} S_P(f) \cdot f_P$.

[2]Note that path inflation (additional delays) is reasonable with DisMR and for delay-sensitive traffic, it can be included as a constraint as well to limit the additional hop/delay. Due to the space limit, we omit those performance comparisons.

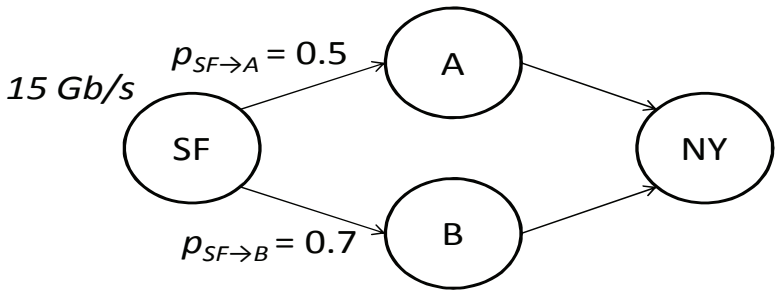


Figure 4.1: Simple 4-node topology: link-capacity=10Gbps, U_{max}=0.9, traffic demand=15Gbps from SF→NY with two multiple paths. Two links are equipped with monitors as sampling rate: $P_{SF\to A} = 0.5$ and $P_{SF\to B} = 0.7$.

3. Link Utilization: This metric reflects TE constraints.

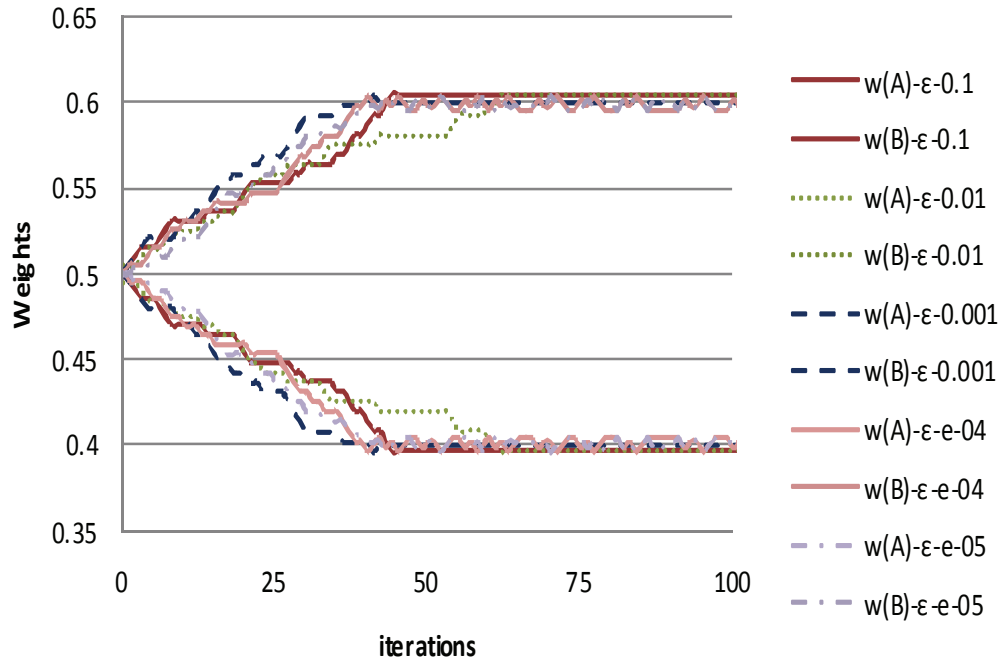
4. Sum of Weights Changes: It is defined as the sum of all weights changes for all routers and quantifies the oscillations observed in the system. .

4.4.2 Sensitivity Analysis and Parameter Tuning

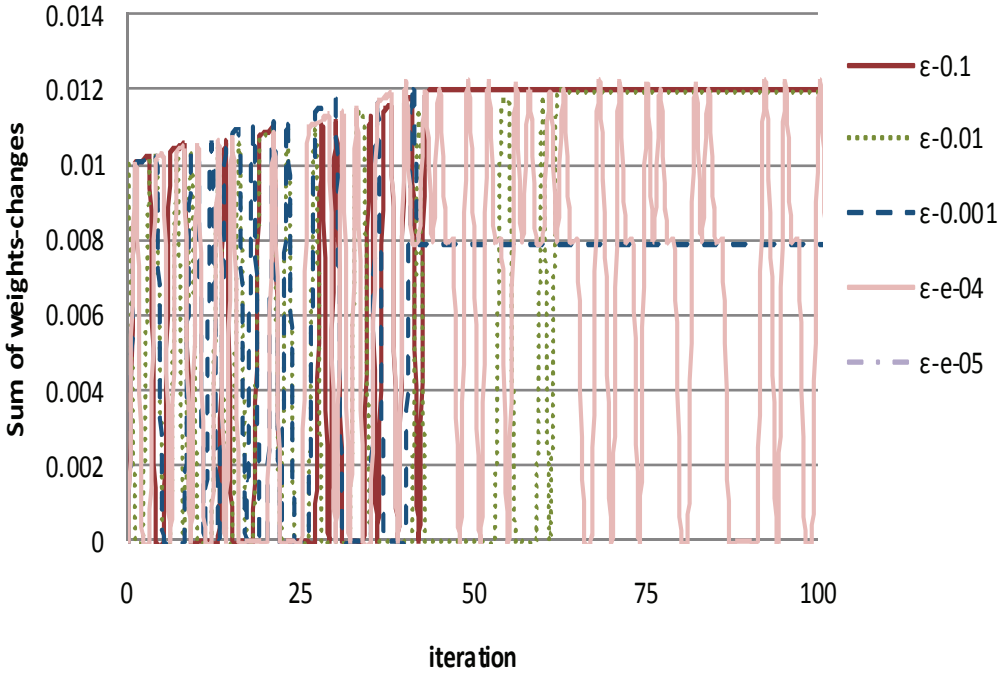
Our simulation results show that our performance is largely independent of α and β. We find $\alpha = \beta = 10^{-5}$ offers a good trade-off between the consequences discussed in [33] and we use them for all of our evaluations. The more sensitive parameters are mostly *migration threshold*, $\epsilon \times m_\zeta$ and *migration rate*, Δ_{fix}. We explore these parameters in the following sections.

Choosing the migration threshold

The *migration threshold* is defined as the product of ϵ and m_ζ. It controls the granularity of equilibrium DisMR wants to achieve. We start with the simple 4-node topology depicted in Figure 4.1 to explore these two parameters. With $U_{max} = 0.9$, the optimal split-ratios are $w(SF, NY, A) = \frac{6}{15}$ and $w(SF, NY, B) = \frac{9}{15}$ with link utilization $U_A = \frac{6}{10} < U_{max}$ and $U_B = \frac{9}{10} \leq U_{max}$. The offline maximum achievable gain calculated by centralized MeasuRouting is $6 \times 0.5 + 9 \times 0.7 = 9.3$ without TE-violation. Table 4.1 shows that the performance of our proposed DisMR algorithm is

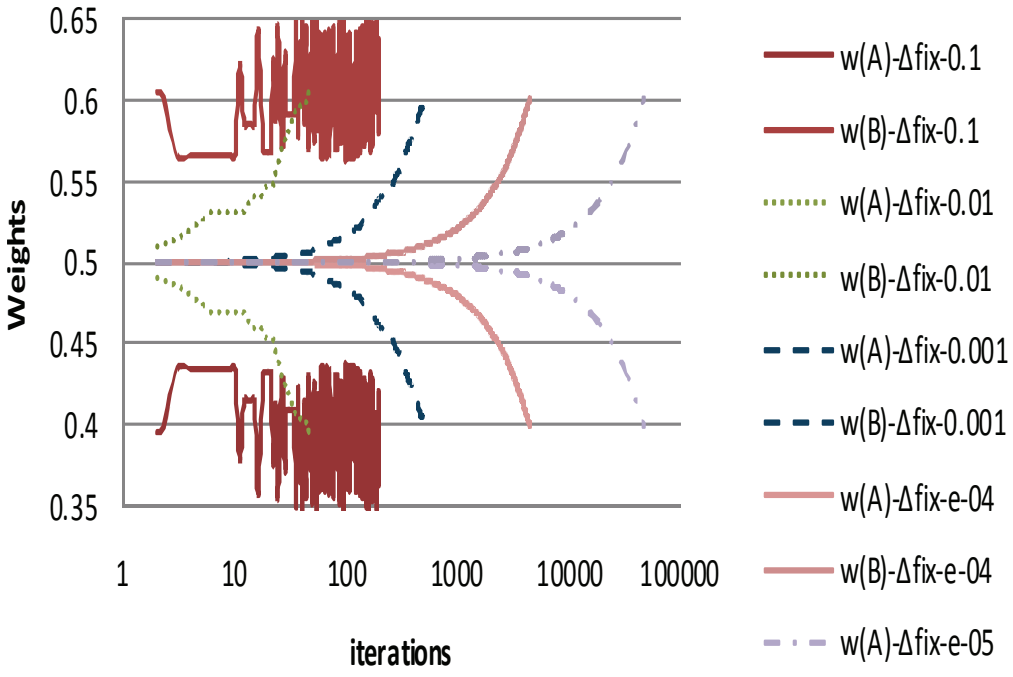


(a) Weights Distribution (ϵ-variations)

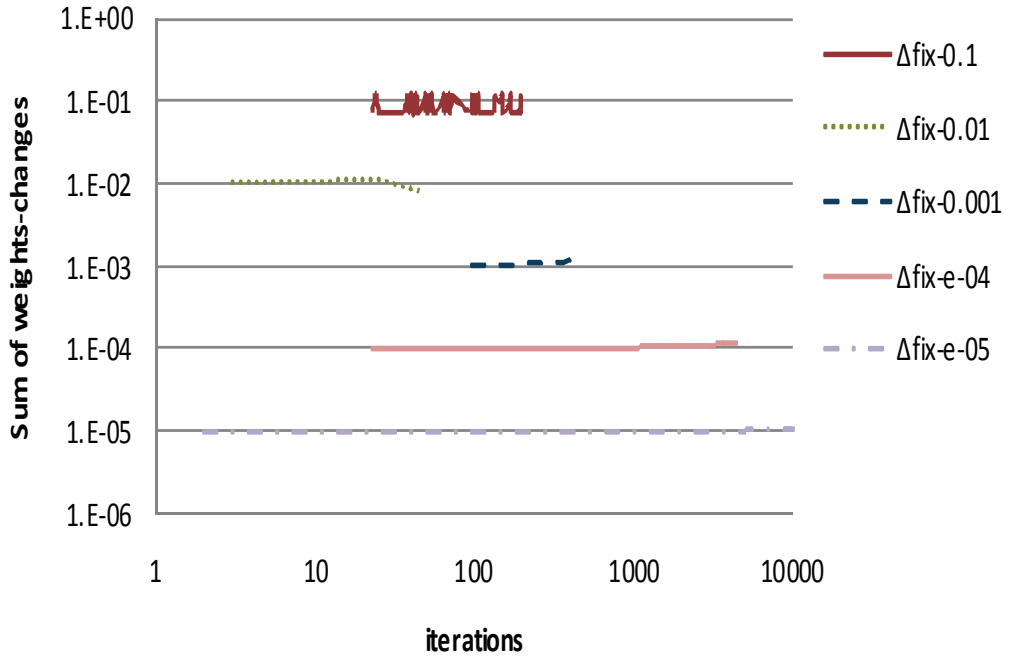


(b) Sum of Weights Changes (ϵ-variations)

Figure 4.2: System performance comparison with ϵ variation



(a) Weights Distribution (Δ_{fix}-variations)



(b) Sum of Weights Changes (Δ_{fix}-variations)

Figure 4.3: System performance comparison with Δ_{fix} variation

Table 4.1: m_ζ variations with $\epsilon = 10^{-3}$, $\Delta_{fix} = 10^{-2}$ in 4-node topology

m_ζ	10^3	10^4	10^5	10^6	10^7
iterations	249	54	49	66	34
U_A	0.5993	0.5999	0.5999	0.5999	0.5999
U_B	0.9007	0.9001	0.9001	0.9001	0.9001
C_A	0.5	0.5	0.5	0.5	0.5
C_B	1.0281	1.7087	14.387	141.17	1408.9
$\epsilon \times m_\zeta$	1	10	100	1000	10000
W_A	0.3995	0.3999	0.3999	0.3999	0.3999
W_B	0.6005	0.6001	0.6001	0.6001	0.6001
$Gain$	9.3006	9.3003	9.3003	9.3003	9.3003

Table 4.2: ϵ variations with $m_\zeta = 10^6$, $\Delta_{fix} = 10^{-2}$ in 4-node topology

ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	Theo.-Equil.
iterations	45	63	43	551	5625	∞
U_A	0.5939	0.5939	0.5998	0.59995	0.599993	0.5999998
U_B	0.90608	0.90608	0.9002	0.90005	0.900006	0.9000002
C_A	0.5	0.5	0.5	0.5	0.5	0.5
C_B	6080.308	6080.308	141.1678	55.1099	6.6940	0.5
$\epsilon \times m_\zeta$	100000	10000	1000	100	10	0
W_A	0.39594	0.39594	0.399906	0.399963	0.3999957	0.39999987
W_B	0.60406	0.60406	0.600093	0.600036	0.6000042	0.60000013
$Gain$	9.31216	9.31216	9.30028	9.300109	9.3000127	9.3000004

less sensitive to the sharpness of the penalty, m_ζ and it has similar gain as centralized MeasuRouting only with subtle TE-violation (e.g., only $\frac{0.0001}{0.9} = 0.001$) except $m_\zeta = 10^3$. We use $m_\zeta = 10^6$ for the following simulations.

With $m_\zeta = 10^6$ and $U_{max} = 0.9$, the theoretical equilibrium (e.g., denoted as "Theo.-Equil." in Table 4.2) will occur in both of the cost metric $\Psi(SF, NY, A) = (1-0.5)+0 = \Psi(SF, NY, B) = (1-0.7)+(\frac{9.000002}{10}-0.9)\cdot 10^6 = 0.3+0.2 = 0.5$ where the weight split-ratios are $w(SF, NY, A) = \frac{5.999998}{15}$ and $w(SF, NY, B) = \frac{9.000002}{15}$ with link utilization $U_A = \frac{5.999998}{10} < U_{max}$ and $U_B = \frac{9.000002}{10} > U_{max}$. Therefore the theoretical measurement gain will be $5.999998 \times 0.5 + 9.000002 \times 0.7 = 9.3000004$ with negligible TE-violation (e.g., only $\frac{0.0000002}{0.9} = 2.2 \cdot 10^{-7}$). However, it requires the finest traffic-migration granularity (e.g., infinitesimal Δ_{fix}) which takes large number of iteration times to converge. Table 4.2 compares the performance of DisMR with

Table 4.3: Δ_{fix} variations with $m_\zeta = 10^6$, $\epsilon = 0.001$ in 4-node topology

Δ_{fix}	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	Theo.-Equil.
iterations	195	48	498	4424	46079	∞
U_A	0.59969	0.59986	0.59947	0.59994	0.59999	0.5999998
U_B	0.90030	0.90014	0.90053	0.90006	0.90001	0.9000002
C_A	0.5	0.5	0.5	0.5	0.5	0.5
C_B	300.4599	141.1678	527.1125	61.0521	5.437467	0.5
$\epsilon \times m_\zeta$	1000	1000	1000	1000	1000	0
W_A	0.39979	0.39991	0.399648	0.3999594	0.3999965	0.39999987
W_B	0.60020	0.600093	0.600351	0.6000405	0.60000342	0.60000013
$Gain$	9.3006	9.30028	9.30105	9.300122	9.30001027	9.3000004

different choices of ϵ. With smaller ϵ, DisMR has less TE-violation but with longer convergence time and more oscillations (e.g., $U_B = 0.900006 \approx U_{max}$ when $\epsilon = 10^{-5}$ but with 5625 iterations). On the other hand, with larger ϵ, it has more TE-violation but with shorter convergence time and less oscillations (e.g., $U_B = 0.90608 > U_{max}$ when $\epsilon = 10^{-1}$ but with 45 iterations). Therefore, choosing the right ϵ is a tradeoff between convergence speed and TE-violations. We suggest using $\epsilon = 10^{-3}$.

Choosing the migration rate

Table 4.3 shows the performance of DisMR with different choices of migration rate, Δ_{fix}. With smaller Δ_{fix}, it has less TE violation but with longer convergence time (e.g., $U_B = 0.90001 \approx U_{max}$ when $\Delta_{fix} = 10^{-5}$ but with 46079 iterations). With larger Δ_{fix}, it has more TE violation but with shorter convergence time (e.g., $U_B = 0.90030 > U_{max}$ when $\Delta_{fix} = 10^{-1}$ but with 195 iterations). Figure 4.3(a) shows how our algorithm adjusts the weights at SF during the evaluation period with Δ_{fix} variation. For $\Delta_{fix} = 10^{-5}$, it converges to the optimal weights smoothly, but rather slowly. On the contrary, for $\Delta_{fix} = 10^{-1}$, it converges to the optimal weights quickly, but rather unevenly. Figure 4.2 and 4.3 compares how our algorithm adjusts the weights at SF during the evaluation period with Δ_{fix} and ϵ variations. As shown in Figure 4.2(b), for smaller ϵ, the system has frequently oscillations. Different from ϵ, for smaller Δ_{fix}, it has smooth oscillation behavior as in Figure 4.3(b). Choosing the right Δ_{fix} is a tradeoff between convergence speed and TE-violations. We suggest using

Table 4.4: Δ_{fix} variations with $m_\zeta = 10^6$, $\epsilon = 0.001$ in Abilene

Δ_{fix}	10^{-1}	$5 \cdot 10^{-2}$	10^{-2}	$5 \cdot 10^{-3}$	10^{-3}
iterations	322	440	3416	7965	23068
TE-violation	$3.524 \cdot 10^{-5}$	$1.062 \cdot 10^{-5}$	$9.236 \cdot 10^{-6}$	$1.33 \cdot 10^{-6}$	$1.10 \cdot 10^{-6}$
$Gain(DisMR)$	2671.9	2671.72	2671.57	2671.54	2671.537
$Gain(Static - MR)$	2671.8	2671.8	2671.8	2671.8	2671.8

Table 4.5: Δ_{fix} variations with $m_\zeta = 10^6$, $\epsilon = 0.001$ in AS6461

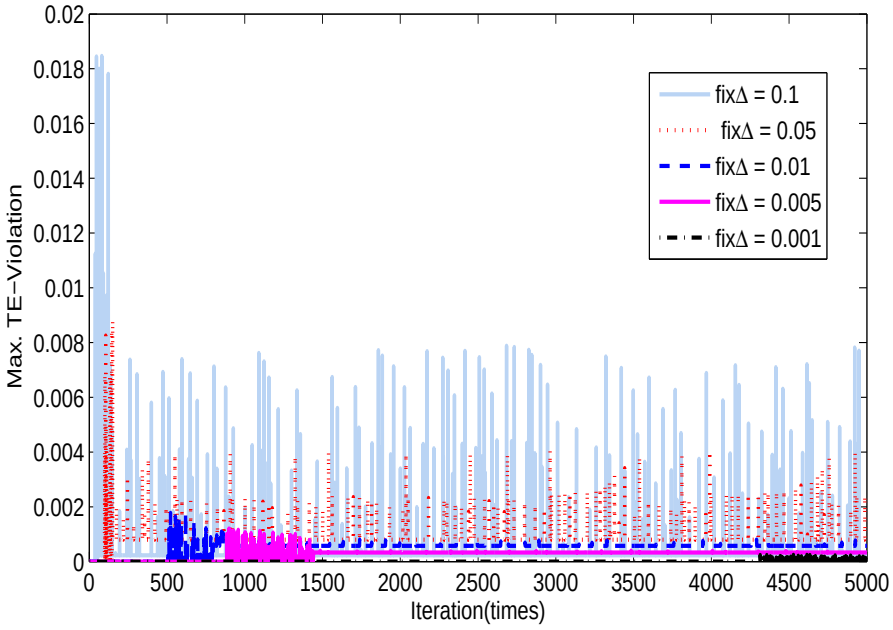
Δ_{fix}	10^{-1}	$5 \cdot 10^{-2}$	10^{-2}	$5 \cdot 10^{-3}$	10^{-3}
iterations	87	119	573	1394	6145
TE-violation	0	0	0	0	0
$Gain(DisMR)$	9648.53	9648.53	9648.53	9648.53	9648.53
$Gain(Static - MR)$	9648.53	9648.53	9648.53	9648.53	9648.53

$\Delta_{fix} = 10^{-2}$.

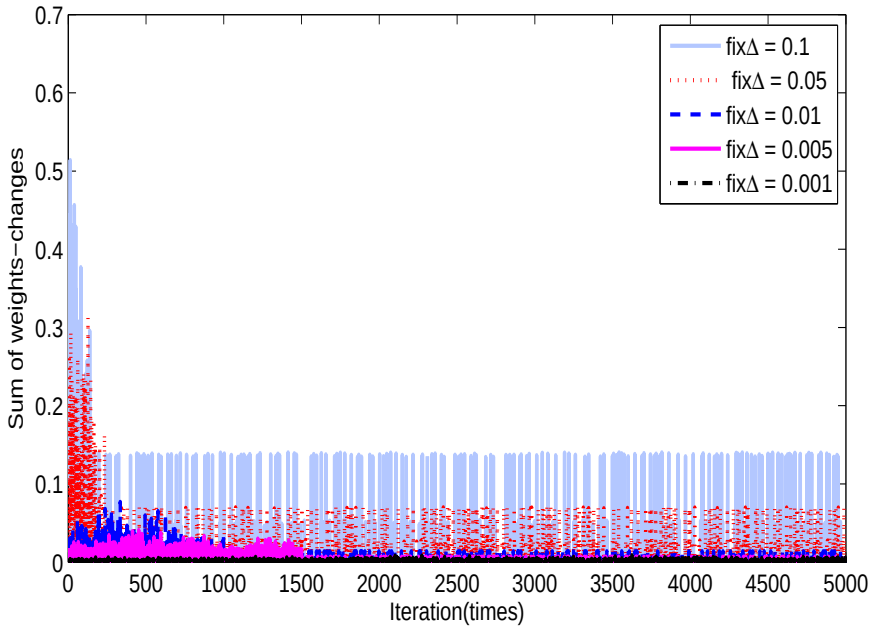
4.4.3 Applied in Realistic Topologies

In this section, we evaluate the performance of DisMR in three realistic topologies, Abilene, GEANT and AS6461. For each topology, we use an Equal Cost Multipath Routing (ECMP)-like algorithm to calculate multi-path routing for all OD pairs. In order to accentuate DisMR's performance, we only consider the traffic traces of the OD pairs with at least two multiple paths. Table 4.4 and 4.5 compares the performance of DisMR with different choices of migration rate, Δ_{fix} in Abilene and AS6461 topologies/traces, where the fixed migration threshold used in this section is 1000 ($\epsilon = 10^{-3}$, $m_\zeta = 10^6$) and the TE-constraint is $U_{max} = 0.9$. Consistent with 4-node topology, DisMR with smaller Δ_{fix} incurs less TE-violation but with longer convergence time, while DisMR with larger Δ_{fix} incurs more TE-violation but with shorter convergence time. The same property could be observed in GEANT network topology.

Figure 4.4 shows the real-time TE-violation and sum of the weights changes of DisMR using GEANT network/trace. The TE-violation value is defined as the absolute value above U_{max}. For $\Delta_{fix} = 10^{-3}$, DisMR has nearly zero TE-violation but converges to the equilibrium state slowly (e.g., 5083 iterations). Also the system oscillation (sum



(a) Max. TE-violation



(b) Sum of Weights Changes per Iteration

Figure 4.4: Δ_{fix} variations in GEANT network topology/trace with $m_\zeta = 10^6$, $\alpha = \beta = 10^{-5}$ and $\epsilon = 10^{-3}$

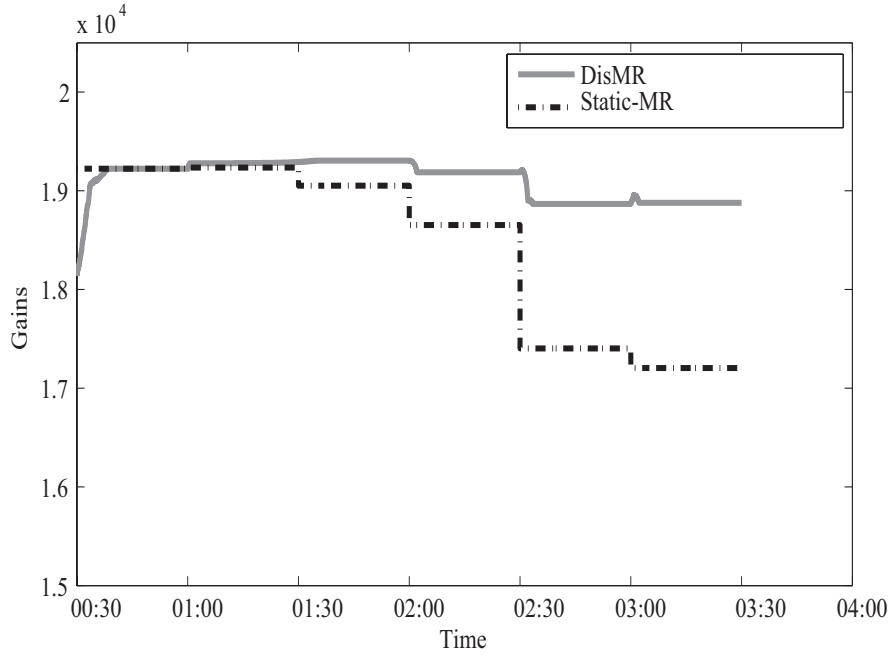
of the weights changes) is quite low (e.g., see Figure 4.4(b)). On the contrary, for $\Delta_{fix} = 5 \cdot 10^{-2}$, DisMR has more TE-violation and higher system oscillation but faster converge time (e.g., 1480 iterations). To conclude, all the simulation results using realistic topologies/traces show that the measurement gain of DisMR is close to the maximum achievable gain using offline, centralized MeasuRouting which is denoted as “Static-MR" in the tables.

4.4.4 Applied in Dynamic Traffic Scenario

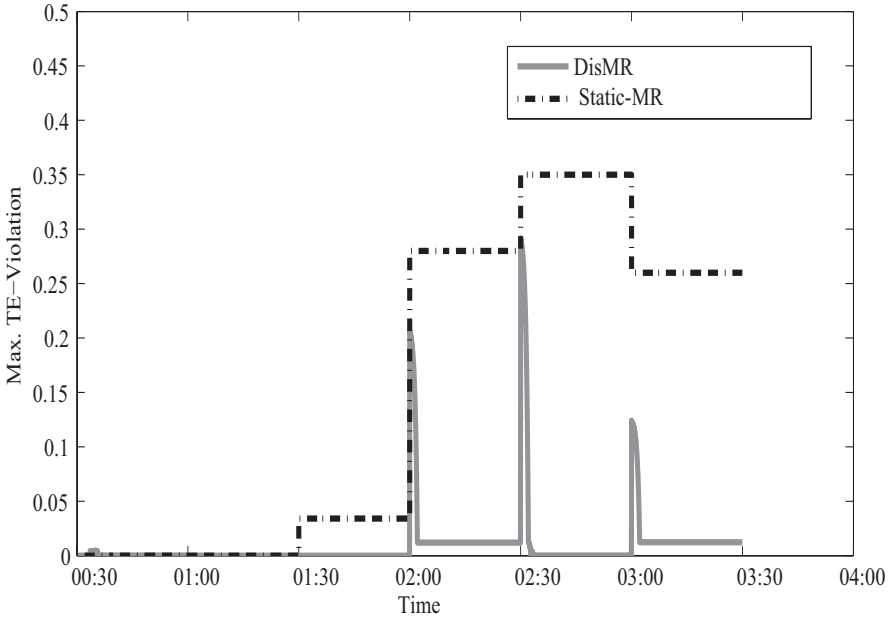
In this section, we compare the performance of DisMR with static centralized MeasuRouting in dynamic traffic scenario. We conducted these experiments using GEANT topology with the traffic snapshots on Apirl 11 and we change the traffic patterns in every 30 minutes based on the traces in [3]. Here Static-MR consistently uses the same traffic splitting strategy based on the initial traffic snapshot (00:30), while DisMR will adaptively adjust its traffic splitting policy with the new traffic pattern. Figure 4.5 shows the real-time $\max$ TE-violations and the changes of measurement gain for DisMR and Static-MR in GEANT network/trace. Initially, DisMR has similar gain as Static-MR after it reaches equilibrium state (00:38) in Figure 4.5(a). We observed that the measurement gain of Static-MR decreases a lot when traffic pattern changed. When the time interval increases (03:30), the degradation becomes severe but DisMR can still outperform Static-MR (e.g., $\frac{1.9-1.7}{1.7} \approx 11.7\%$). In Figure 4.5(b), both DisMR and Static-MR have large TE-violation when the traffic suddenly changes but DisMR can quickly improve its TE-violation in short period of time compared to Static-MR (e.g., up to $\frac{0.35}{0.003} \approx 100X$ at time (03:00)). In brief, DisMR has improved higher measurement gains and much lower TE-violations compared to static, centralized MeasuRouting in dynamic traffic scenario.

4.5 Related Work

Previous work on network-wide traffic measurement mostly focused on solving monitor placement problem for fixed traffic characteristics/monitoring objectives [15, 20, 72]. [20] defines utility functions for the sampled traffic, and maximize the overall



(a) Measurement Gain Comparison



(b) Max. TE-violation Comparison

Figure 4.5: Dynamic traffic scenario

utility with limited operation/deployment cost. [15] improves upon [20] by performing a more rigorous analysis to indicate the convergence of any heuristic solution. Research efforts also exist that implicitly involve distributed monitor placement, so as to detect cases such as DDoS [9], SLA [68] and global iceberg [77]. [65] proposes Successive c-Optimal design to optimize the deployment and sampling rate of large IP networks, in order to better estimate traffic matrix.

Most recently, MeasuRouting paradigm [63] was proposed to assist traffic monitoring for dynamically-changing traffic characteristics by intelligently re-routing interested traffic sub-populations over the pre-deployed monitors. However it is modeled as linear programming problem, and it requires the existence of centralized controller and offline analysis to find the optimal re-routing strategies for every router, which is unpractical in production IP networks. We introduce selfish routing into MeasuRouting problem and define a new routing game based on a novel cost function to de-centralize MeasuRouting. Previously, REPLEX [32], TeXCP [46] and MATE [28] have been proposed as dynamic TE solutions to minimize the path latency or the link utilization by adjusting the split ratios of traffic among the paths with the same ingress/egress nodes. These algorithms take advantage of multiple paths in a network (e.g., MPLS/ECMP multiple paths). In contrast to them, our introduced link cost function is a novel combination of link measurement ability and TE constraint. Moreover our path cost is defined as the product of link costs instead of traditional summation operation.

4.6 Conclusion and Future Work

In this chapter we propose a distributed measurement-aware traffic engineering protocol, DisMR, based on game-theoretic rerouting policy. It achieves the decent balance between measurement-aware routing and traffic engineering objectives by the introduction of a new routing game and distributed routing control. We show that DisMR guarantees both a provable Nash equilibrium and a fast convergence without significant oscillations. The measurement gain of DisMR at the equilibrium state is close to the maximum achievable gain calculated by offline/centralized MeasuRouting

in static traffic case. DisMR also improves the measurement gain and TE-violations of MeasuRouting in dynamic traffic scenario. We plan to perform further simulation-based analysis on other large networks topologies since the propagation delay of sub-path cost information might influence the convergence of DisMR. We also tend to explore different kinds of penalty functions in DisMR to minimize the POA in theory. We would like to study the impacts of asynchronous routing-updates of link/path costs which could be solved similarly as in [28]. Future work also involves extending DisMR to support flows with different measurement/sampling importance.

Chapter 4, in part, is a reprint of the material as it appears in in the following publications:

- Chia-Wei Chang, Han Liu, Guanyao Huang, Bill Lin, and Chen-Nee Chuah, "Distributed Measurement-Aware Routing: Striking a Balance between Measurement and Traffic Engineering", *IEEE 31st International Conference on Computer Communications (INFOCOM)*, Orlando, Florida, March 25-30, 2012.

Chapter 4, in full, has been submitted for publication of material as it may appear in IEEE Transactions on Networking (ToM), Chia-Wei Chang, Han Liu, Guanyao Huang, Bill Lin and Chen-Nee Chuah, " Distributed Measurement-Aware Routing for Multiple Classes of Flows". The dissertation author was the primary investigator and author of the papers.

Chapter 5

Conclusion and Future Work

The network-wide traffic monitoring era has reached a phase where efficient gain-driven routing-assisted monitoring mechanisms are inevitable for network operators to facilitate fine-grained flow-level measurements (e.g., [17–19, 23]), coupled with the fast-changing Internet traffic landscape and large traffic volume. This dissertation tackles the critical problem of gain-driven routing-assisted monitoring mechanisms design especially suited for network-wide traffic monitoring applications where maximizing the overall traffic monitoring utility is our primary design objective. We propose two different approaches to achieve this goal, first, the centralized approach, by solving two-step optimization problems to find the best monitor placement and dynamic routing strategy to achieve maximum measurement utility of the network and then distribute the measurement workload across participated monitors without compromising on the overall traffic measurement gain of the network by using disjoint flow sampling. Second, the distributed method, by proposing a measurement-aware traffic engineering protocol based on a game-theoretic re-routing policy that attempts to optimally utilize existing monitor locations for maximizing the traffic measurement gain of the network while ensuring that the traffic load distribution across the network satisfies some traffic engineering constraint. A novel cost function that reflects both the measurement capabilities of monitors (e.g., packet sampling rates) and the traffic engineering (TE) constraint on each link is applied. Both these approaches provide substantial improvements in traffic measurement gain of the network and load-balanced measurement capability across monitors over existing solutions.

A key challenge in exploring and evaluating our proposed routing-assisted monitoring mechanisms is to accurately model the traffic that needs to be measured by the network operator. The traffic is further divided into sub-populations (e.g., flowset) and we need to differentially route them for measurement purpose based on their relative importance and the measurement capabilities (e.g., packet sampling rates) of monitoring devices deployed on the links. We study our routing-assisted monitoring mechanisms with flows whose sampling importance and sizes are synthetically generated. Each flow is assigned to a flowset and all flows within the same flowset have the identical routing strategy (i.e, same routes). If each flow is assigned to a unique flowset, we can have the greatest degree of freedom to differentially route the traffic in finest granularity. This will allow each flow to be routed independently to maximize the measurement gain of the network. However, it is not scalable from both computational and implementation perspectives. In this thesis, we only have q flowsets per OD-pair traffic. Assume each OD-pair traffic has L flows where $L>q$ and thus each flow is assigned to one of its q flowsets. There can be multiple ways of making such an assignment. Here we assign an equal number of flows to each of the q flowsets per OD-pair. For the traffic demand (size) of each flowset, we first generate aggregate traffic demand for each OD-pair by using Gravity Model [55] and then divided by q. It was observed that the way traffic aggregates (e.g., OD-pair traffic) are decomposed into several traffic sub-populations (e.g., flowsets) has an impact on the performance of routing-assisted traffic measurement mechanisms. The same observation could be also found in [63].

The properties of flow importance for each flowset in this thesis are modeled in a very generic way such that our framework can be applied to a wide variety of measurement scenarios. We assume that the dynamic traffic/measurement changes will stay for long time enough for us to re-optimize monitor placement and flowset routing in centralized MMPR case[1]. Therefore we manipulate the flow importance in flowset i to be greater than the flowset i+1 per OD-pair traffic where $i = 1 \cdots (q\text{-}1)$. We also observed that the diversity of flow importance per OD-pair has a relation to the performance of our proposed centralized MMPR framework: the measurement gain increases tremendously if the diversity of flow importance also increases. We plan to

[1]We later relax this assumption in distributed routing-assisted monitoring mechanism case.

explore such phenomenon in much greater detail. Meanwhile, how to define proper measurement gain function to reflect different measurement tasks/applications remains open problem (i.e., here we simply use flow importance factor $\times$ flow size) .

The other important factor to affect the performance of our proposed routing-assisted traffic monitoring mechanisms is the routing strategy/protocol determined to use. For example, a routing protocol that strictly routes traffic between an OD-pair traffic along only with the equal cost multi-path (ECMP) may provide less opportunities to re-route important flows through monitors and resulted in poor measurement gain improvement. In our centralized routing-assisted approach (MMPR), the centralized routing controller, is able to detour flows away from the shortest path as long as it works within the constraints of existing intra-domain traffic engineering (TE) operations. Such dynamic forwarding mechanism can be implemented using programmable routers [13, 57, 59]. Besides this, how to dynamically configure routing table entries is also important. For our distributed approach (DisMR), we assume synchronized routing-updates of link/path cost information. This information exchange resembles a distance vector routing protocol. The impacts of asynchronous update issue could be solved similarly in [28] where we defer as our future work. Furthermore, it would be interesting to treat sampling rate as another degree of freedom (e.g., as proposed in [15, 20]), to let monitors dynamically adjust them depending on the amount of traffic passing through. For the distributed approach, these information can be reflected in our proposed novel cost function to drive flows to be attracted to the links with better measurement capabilities (e.g., higher packet sampling rates) while avoiding TE violations. We also tend to explore different kinds of penalty functions which are used to reflect TE violations to get more similar traffic measurement gain of the network as centralized MeasuRouting approach (i.e., minimize the price of anarchy(POA) in theory). Finally, the implementation of both our proposed centralized and distributed routing-assisted traffic monitoring mechanisms can be leveraged by using OpenFlow [59] or any other programmable routing platform.

Bibliography

[1] The abilene network. http://www.internet2.edu.

[2] Abilene traffic matrices. http://www.cs.utexas.edu/ yzhang/research/AbileneTM.

[3] Geant topology. http://archive.geant.net/upload/pdf/Topology_Oct_2004.pdf.

[4] Ibm cplex software. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[5] Totem, a toolbox for traffic engineering methods, february 2005. http://totem.info.ucl.ac.be.

[6] Totem geant traffic matrices. http://totem.info.ucl.ac.be/dataset.html.

[7] Cisco netflow. http://www.cisco.com/, 2009.

[8] D. Antoniades, M. Polychronakis, S. Antonatos, E. P. Markatos, S. Ubik, and A. Oslebo. Appmon: An application for accurate per application traffic characterization. In *Proceedings of IST Broadband Europe 2006 Conference*, 2006.

[9] P. E. Ayres, H. S., H. J. Chao, and W. C. Lau. ALPi: A DDoS Defense System for High-Speed Networks. *IEEE Journal on Selected Areas in Communications*, 24(10):1864–1876, 2006.

[10] H. Ballani and P. Francis. CONMan: A Step Towards Network Manageability. In *Proceedings of ACM SIGCOMM*, 2007.

[11] M. Beckmann, C. B. McGuire, and C. B. Winsten. Studies in the Economics of Transportation. Yale University Press 1956.

[12] Y. Bejerano and R. Rastogi. Robust monitoring of link delays and faults in IP networks. In *Proceedings of IEEE INFOCOM*, April 2003.

[13] C. Wiseman and J. Turner and M. Becchi and P. Crowley and J. Dehart and M. Haitjema and S. James and F. Kuhns and J. Lu and J. Parwatikar and R. Patney and M. Wilson and K. Wong and D. Zar. A remotely accessible network

processor-based router for network experimentation. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems* , ANCS'08, 2008.

[14] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. V. D. Merwe. Design and implementation of a Routing Control Platform. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, 2005.

[15] G. R. Cantieni, G. Iannaccone, C. Barakat, C. Diot, and P. Thiran. Reformulating the monitor placement problem: Optimal network-wide sampling. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies*, CoNEXT'06, 2006.

[16] V. Chan. Near-term future of the optical network in question? *IEEE Journal on Selected Areas in Communication*, 25(9):2, December 2007.

[17] C.-W. Chang, A. Gerber, B. Lin, S. Sen, and O. Spatscheck. Network DVR: A Programmable Framework for Application-Aware Trace Collection. In *Proceedings of ACM Passive and Active Measurement Conference*, PAM'10, 2010.

[18] C.-W. Chang, S. Lee, B. Lin, and J. Wang. The Taming of The Shrew: Mitigating Low-Rate TCP-Targeted Attack. In *Proceedings of IEEE 29th International Conference on Distributed Computing Systems(ICDCS)*, June 2009.

[19] C.-W. Chang, S. Lee, B. Lin, and J. Wang. The Taming of The Shrew: Mitigating Low-Rate TCP-Targeted Attack. *IEEE Transactions on Network and Service Management (TNSM)*, 7(1):1–13, March 2010.

[20] C. Chaudet, E. Fleury, I.G. Lassous, H. Rivano, and M-E. Voge. Optimal positioning of active and passive monitoring devices. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies* , CoNEXT'05, pages 71–82, 2005.

[21] B.-Y. Choi and S. Bhattacharyya. On the Accuracy and Overhead of Cisco Sampled NetFlow. In *Proceedings of ACM SIGMETRICS Workshop on Large Scale Network Inference (LSNI)*, June 2005.

[22] K.-C. Claffy, G.-C. Polyzos, and H.-W. Braun. Application of Sampling Methodologies to Network Traffic Characterization. In *Proceedings of ACM SIGCOMM*, 1993.

[23] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: a stream database for network applications. In *Proceedings of ACM Sepecial Interest Group On Management Of Data (SIGMOD)*, 2003.

[24] S. C. Dafermos and F. T. Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards*, 73B(2):91–118, 1969.

[25] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood. Deep Packet Inspection Using Parallel Bloom Filters. In *IEEE Micro*, pages 44–51, 2003.

[26] N. Duffield and M. Grossglauser. Trajectory Sampling for Direct Traffic Observation. In *Proceedings of ACM SIGCOMM*, 2001.

[27] K. Elias and P. Christos. Worst-case equilibria. In *Proceedings of the 16th annual conference on Theoretical aspects of computer science*, STACS'99, 1999.

[28] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings of IEEE INFOCOM*, 2001.

[29] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a Better NetFlow. In *Proceedings of ACM SIGCOMM*, 2004.

[30] C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, 21(3):270–313, August 2003.

[31] A. Feldmann, A. G. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving Traffic Demands for Operational IP Networks: Methodology and Experience. In *Proceedings of ACM SIGCOMM*, 2000.

[32] S. Fischer, N. Kammenhuber, and A. Feldmann. REPLEX - Dynamic Traffic Engineering Based on Wardrop Routing Policies. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies*, CoNEXT'06, Dec 2006.

[33] S. Fischer, H. R*ä*cke, and B. V*ö*cking. Fast convergence to Wardrop equilibria by adaptive sampling methods. In *Proceedings of 38th ACM Symposium on Theory of Computing*, STOC'06, May 2006.

[34] B. Fortz and M. Thorup. Internet TE by Optimizing OSPF Weights. In *Proceedings of IEEE INFOCOM*, March 2000.

[35] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery. In *Proceedings of IEEE INFOCOM*, April 2000.

[36] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Meyers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A Clean Slate 4D Approach to Network Control and Management. *ACM SIGCOMM Computer Communication Review*, 35(5), October 2005.

[37] N. Hohn and D. Vetch. Inverting Sampled Traffic. In *Proceedings of ACM Internet Measurement Conference*, November 2003.

[38] J. Horton and A. L. Ortiz. On the number of distributed measurement points for network tomography. In *Proceedings of ACM Internet Measurement Conference*, November 2003.

[39] G. Huang, C.-W. Chang, C.-N. Chuah, and B. Lin. Measurement-aware Monitor Placement and Routing: A Joint Optimization Approach for Network-Wide Measurements. *IEEE Transactions on Network and Service Management (TNSM)*, accepted to appear in 2011.

[40] G. Huang, A. Lall, C-N. Chuah, and J. Xu. Uncovering Global Icebergs in Distributed Monitors. In *Proceedings of IEEE International Workshop on Quality of Service* , IWQoS'09, July 2009.

[41] G. Huang, S. Raza, S. Seetharaman, and C.-N. Chuah. Dynamic Measurement-Aware Routing in Practice. *IEEE Networks*, 25(3):29–34, 2011.

[42] M. Iliofotou, M. Faloutsos, and M. Mitzenmacher. Exploiting Dynamicity in Graph-based Traffic Analysis: Techniques and Applications. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies*, CoNEXT'09, 2009.

[43] IS-IS. http://tools.ietf.org/html/rfc1142, February 1990.

[44] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring TCP Connection Characteristics Through Passive Measurements. In *Proceedings of IEEE INFOCOM*, March 2004.

[45] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of internet instrumentation. In *Proceedings of IEEE INFOCOM*, April 2000.

[46] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: responsive yet stable traffic engineering. In *Proceedings of ACM SIGCOMM*, 2005.

[47] A. Kumar, M. Sung, J. Xu, and J. Wang. Data streaming algorithms for efficient and accurate estimation of flow size distribution. In *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems* , SIGMETRICS'04, pages 177–188, 2004.

[48] A. Lakhina, M. Crovella, and C. Diot. Diagnosing Network-Wide Traffic Anomalies. In *Proceedings of ACM SIGCOMM*, 2004.

[49] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft. Structural Analysis of Network Traffic Flows. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, SIGMETRICS '04, 2004.

[50] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina. Detection and Identification of Network Anomalies Using Sketch Subspaces. In *Proceedings of ACM Internet Measurement Conference*, IMC'06, 2006.

[51] X. Li, F. Bian, H. Zhang, C. Diot, R. Govindan, W. Hong, and G. Iannaccone. MIND: A Distributed Multidimensional Indexing for Network Diagnosis. In *Proceedings of IEEE INFOCOM*, April 2006.

[52] P. Loiseau, P. Gonçalves, S. Girard, F. Forbes, and P.-V.-B. Primet. Maximum likelihood estimation of the flow size distribution tail index from sampled packet data. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, SIGMETRICS '09, 2009.

[53] H.-V. Madhyastha and B. Krishnamurthy. A Generic Language for Application-Specific Flow Sampling. *ACM SIGCOMM Computer Communication Review*, 38(2), April 2008.

[54] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang. Is sampled data sufficient for anomaly detection? In *Proceedings of ACM Internet Measurement Conference*, IMC'06, 2006.

[55] A. Medina, N. Taft, S. Battacharya, C. Diot, and K. Salamatian. Traffic matrix estimation: Existing techniques compared and new directions. In *Proceedings of ACM SIGCOMM*, August 2002.

[56] D. Moore, G. M. Voelker, and S. Savage. Inferring Internet Denial of Service Activity. In *Proceedings of the 10th USENIX Security Symposium*, August 2001.

[57] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles* , SOSP, pages 217–231, December 1999.

[58] H. Nguyen and P. Thiran. Active measurement for multiple link failures diagnosis in ip networks. In *ACM Passive and Active Measurement Conference*, Proceedings of PAM Workshop, 2004.

[59] The OpenFlow Switch Consortium. http://www.openflowswitch.org.

[60] OSPF. http://tools.ietf.org/html/rfc2328, April 1998.

[61] Sampling and Filtering Techniques for IP Packet Selection. Internet Draft draft-ietf-psamp-sample-tech-11.txt, Work in Progress, July 2008.

[62] A. Ramachandran, S. Seetharaman, N. Feamster, and V. Vazirani. Fast Monitoring of Traffic Subpopulations. In *Proceedings of ACM Internet Measurement Conference*, IMC'08, 2008.

[63] S. Raza, G. Y. Huang, C-N. Chuah, S. Seetharaman, and J. P. Singh. MeasuRouting: A Framework for Routing-Assisted Traffic Monitoring. In *Proceedings of IEEE INFOCOM*, March 2010.

[64] M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proceedings of the 1999 USENIX LISA Systems Administration Conference*, 1999.

[65] G. Sagnol, M. Bouhtou, and S. Gaubert. Successive c-optimal designs: A scalable technique to optimize the measurements on large networks, 2010.

[66] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen. CSAMP: A System for Network-Wide Flow Monitoring. In *Proceedings of USENIX Symposium on Networked Systems Design & Implementation*, NSDI'08, 2008.

[67] M. R. Sharma and J. W. Byers. Scalable Coordination Techniques for Distributed Network Monitoring. In *Proceedings of ACM Passive and Active Measurement Conference*, PAM'05, April 2005.

[68] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and efficient SLA compliance monitoring. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2007.

[69] N. Spring, R. Mahajan, and T. Anderson. Quantifying the Causes of Path Inflation. In *Proceedings of ACM SIGCOMM*, 2003.

[70] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Network Topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM*, 2002.

[71] A. Sridharan and T. Ye. Tracking Port Scanners on the IP Backbone. In *Proceedings of ACM SIGCOMM*, 2007.

[72] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors: Complexity, heuristics and coverage. In *Proceedings of IEEE INFOCOM*, March 2005.

[73] M. Yoo, C. Qiao, and S. Dixit. Optical burst switching for service differentiation in the next-generation optical Internet. *IEEE Communication Magazine*, 39(2):98–104, 2001.

[74] F. Yu, R.-H. Katz, and T.-V. Lakshman. Gigabit Rate Packet Pattern-Matching Using TCAM. In *Proceedings of IEEE International Conference on Network Protocols*, ICNP'04, pages 174–183, 2004.

[75] L. Yuan, C.-N. Chuah, and P. Mohapatra. ProgME: Towards Programmable Network MEasurement. In *Proceedings of ACM SIGCOMM*, 2007.

[76] Y. Zhang, M. Roughan, N. Duffield, , and A. Greenberg. Fast Accurate Computation of Large-scale IP Traffic Matrices from Link Loads. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, SIGMETRICS'03, 2003.

[77] Q. Zhao, M. Ogihara, H. Wang, and J. Xu. Finding global icebergs over distributed data sets. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, 2006.