Improving Network Performance through Measurement Based Analysis

A Dissertation Presented

by

Rupa Krishnan

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2010

Stony Brook University

The Graduate School

Rupa Krishnan

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of this dissertation.

Dr. Jie Gao, Dissertation Advisor
Assistant Professor, Computer Science Department

Dr. Tzi-cker Chiueh, Chairperson of Defense
Professor, Computer Science Department

Dr. Samir Das
Professor, Computer Science Department

Dr. Jennifer Wong
Assistant Professor, Computer Science Department

Dr. Rui Zhang-Shen
Software Engineer, Google Inc.

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Improving Network Performance through Measurement Based Analysis

by

Rupa Krishnan

Doctor of Philosophy

in

Computer Science

Stony Brook University

2010

The unprecedented increase in the scale of established networks such as the Internet and Cellular Networks and evolution of new kinds of networks such as Wireless Mesh Networks, Sensor Networks, etc., has led to several challenging research problems in maintaining and improving network performance.

In this dissertation I study some of the challenges in the design and evaluation of routing protocols and examine how performance problems of existing routing algorithms can be understood and solved. I focus on how routing affects performance in two networks - the Content Distribution Networks in the Internet and Wireless Mesh Networks, which represent widely different design considerations. In both networks, there are many factors that affect the network performance. For the Internet, the routing policies, network management, traffic pattern, network dynamics are all important influential factors. In wireless mesh networks, the wireless channel characteristics, interference, environmental changes can also affect the system performance. Thus theoretical analysis considering all these factors are prohibitively difficult. I take a measurement based approach and through extensive measurements and observations derive effective mechanisms to improve their real world performance.

To understand and analyze how routing affects the performance of Content Distribution Networks in the Internet we have developed a tool, called *WhyHigh* to diagnose the cause for inflated latency on any given path. We have used *WhyHigh* to diagnose several instances of inflated latencies, and our efforts over the course of a year have significantly helped improve client performance of a large CDN. Accurate location information for nodes in the network is important in diagnosing inflated paths, hence we also examine causes of errors in localization algorithms and propose heuristics for improving the quality of localization.

For wireless mesh networks, we have developed a Channel Characteristics-Aware Routing Protocol (CARP) that uses measurement based analysis to model and exploit characteristics of wireless channels. We identify better paths even when the channel experiences high temporal fluctuation. We have empirically quantified the performance gain of different protocol mechanisms on a multi-hop wireless test-bed called MiNT-m.

In summary, as networks scale and the networked system becomes extremely complex, theoretical models are often limited in applicability due to the large number of input parameters in a real system. Measurement based analysis, diagnosis and protocol development are complementary to theoretical analysis. This dissertation demonstrate the effectiveness of measurement based approaches in both wired and wireless network settings.

Dedicated to Mom, Dad, Ramya, my husband Sujit and the little one who will be here soon.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Routing is one of the most important components of any communication infrastructure. The routing layer is responsible for managing a host of functionalities such as connectivity, resilience to failure, load balancing, and end-user performance, for the entire network. As a result routing algorithms have been researched extensively since the 1970s. However, routing is still a significant research problem. This is partially due to the evolution of new kinds of networks such as Wireless Mesh networks, Sensor Networks, etc. resulting in different design considerations. It is has also gained prominence due to the unprecedented increase in the scale of networks such as the Internet and Cellular Networks. Initially routing protocols were mainly responsible for maintaining connectivity and load balancing. However performance of the network in terms of bandwidth and latency has recently gained importance for several classes of networks. Hence routing algorithms of today are also responsible for choosing routes which maximize client performance. My objective in this thesis is to study some of the challenges in the design and evaluation of routing protocols and examine how performance problems of existing routing algorithms can be better understood and solved. I also focus on identifying good measurement primitives and platforms for identifying performance issues in existing networks.

The Internet is a network of networks with thousands of users under different administrative domains. With the advent of the world-wide web, social networking, and online video websites such as youtube Internet has become increasingly content driven and performance sensitive. In this work we examine how routing in the Internet can affect performance in terms of latency for users of Content Distribution networks. Identifying performance problems on the Internet is a challenging task due to its distributed nature. One important aspect of troubleshooting is to identify geographical locations of end users and routers. In this dissertation we also present techniques to infer node locations from network measurements. We show that using extensive diagnostic information we can identify clients experiencing inflated latencies. It is important to note that due to its inherent nature, modifications to Internet routing are hard to implement and changes can typically only be effected at the policy level of an administrative domain.

I also examine Wireless Mesh networks (WMN) which are emerging as popular technology for last mile access. Bandwidth and latency are particularly important considerations in WMNs. Interference from within the network and from other wireless sources such as microwaves and other wireless networks can have significant impact on the performance of different network paths, and all layers of the networking stack including routing have to take this into account. Emerging networks allow greater flexibility in the design of routing algorithms. To maximize performance, routing algorithms can be designed to ensure that flows are routed such that they minimize intra network interference and maximize user bandwidth.

Using these two examples - Content Distribution Networks and Wireless Mesh Networks, I examine two case studies of network systems and the role of routing protocols therein. These two networks represent widely different design considerations. However, they give a good overview of the spectrum of roles a routing protocol is expected to fit in. In terms of achieving our objective of understanding and evaluating routing performance both these networks pose very different challenges. For example, the parameters for measuring performance are very different. While reliability of the network maybe an important goal for an ISP in the Internet, a Wireless Mesh Network administrator may want to maximize throughput and reduce interference in order to give better service to clients.

The approach used for evaluating protocols in these settings is based on extensive measurements. For each case we have used extensive evaluation to understand and improve routing algorithms.

1.1 Importance of Empirical Analysis

There has been significant work in understanding theoretical aspects of routing algorithms. Theoretical analysis typically involves making certain simplifying assumptions to make the problem tractable. However, in the real world these assumptions are not always valid, and hence may not reflect routing algorithm behaviour accurately. Examples of such simplifying assumptions are assuming that nodes in the network are uniformly randomly distributed and have uniformly distributed degrees. Real world networks are often severely skewed based on where they are deployed and hence this assumption may not expose all the problems of such deployments. Other simplifying assumptions such as assuming unit-disk signal distribution in wireless networks, or assuming complete information in the Internet would not be valid in the real world.

In the Internet, there has been work on convergence and stability of BGP [34, 95, 101]. However, since BGP is based on configurations created by human operators, and we typically do not have a global view of ISP policies, modelling BGP behaviour under all possible real-world scenarios is not feasible. One can at best perform static analysis of policies from the perspective of a single ISP [30]. However this kind of analysis does not detect cases of poor paths since they are also affected by how much traffic is routed along different paths. Hence, we take a measurement based approach of using existing measurement infrastructure in the Internet to analyze BGP performance in the real-world.

In wireless networks, theoretical studies have focused on modeling the wireless channel [38,47,86]. Although they provide good asymptotic bounds for worst case behaviour, they fail to account for all the factors affecting the wireless channel in the real world [4,22]. Hence, we use wireless test-beds as evaluation platforms to evaluate wireless protocols under a variety of real-world channel conditions.

Measurement based studies face several challenges. In large scale networks such as the Internet getting access to measurements is hard since we cannot access individual routers. One can either have an end-user view of the Internet or at most the view of a particular ISP. Therefore, any analysis on this data must be made from incomplete and noisy datasets. For example using tools such as TRACEROUTE one can only infer the forward path from a server to a client which does not help in identifying problems along the reverse path. In such cases hints in the available dataset (such as time to live fields in TRACEROUTES) can be used to detect inconsistent information. Also the quality of information fed to different algorithms is important. For example in localization of internet hosts adding additional incorrect constraints can significantly affect the performance of the algorithm. Hence measurement based studies also need to keep in mind the kind of data needed as input to different algorithms.

In wireless networks one can safely assume that one has access to all the nodes since they are typically under a single administrative domain. However, the wireless channel is extremely unpredictable since it is open to interference, and is affected by physical obstacles and environmental conditions. As a result wireless link quality experiences significant temporal variations. Consequently, even a comprehensive set of measurements does not make analysis easy and theoretical models do not come close to predicting channel conditions. Based on this we conclude that we not only need good measurement and evaluation infrastructure, also the algorithms themselves should be capable of leveraging and adjusting to channel conditions.

Several frameworks have been proposed which have provided valuable information for analyzing network performance. Planetlab [2] is a very popular infrastructure that allows researchers to make end-to-end measurements at the Internet scale, by allowing users access to machines along the edge of the network. Routeviews [75] is another project which gives researchers access to BGP tables from a set of routers in the Internet. These works help us piece together a better picture of routing in the Internet.

Similarly wireless test-beds such as Roofnet [4], MiNT-m [21], and ORBIT [87] have provided platforms for evaluating wireless routing protocols under different conditions to better understand their behaviour. In this thesis we have made extensive use of empirical analysis based on different measurement frameworks and evaluated our algorithms in real world settings.

1.2 Thesis Contributions

In this dissertation I present several contributions that improve state of the art in identifying performance problems that can be attributed to routing. I also propose novel solutions to alleviate these problems.

1.2.1 Troubleshooting Path Latencies In A Large Content Distribution Network

The Internet is a network of networks, with more than 500 million hosts [44] in 24,000 [17] different administrative domains or Autonomous Systems (AS). Routing in such a large scale system is extremely complex. Also because of its architecture the main considerations in routing are often economical. Different administrative domains want to route packets such that they incur less expense and maximize the utilization of their network. This could lead to the side-effect that some clients may suffer significantly higher latencies or experience lower throughputs. Also since packets typically cross multiple administrative domains implementing a policy in one domain could affect all others along that path.

Path performance is especially significant in Content Distribution Networks since the administrators want to minimize user perceived latency to their content. To address this problem we have developed a tool called called *WhyHigh* that diagnoses the cause for inflated latency on any given path. *WhyHigh* also allows administrators of CDNs to evaluate the performance of different servers in terms of client latency and identify specific causes of poor performance. We have used *WhyHigh* to diagnose several instances of path inflation, and suggest changes to the BGP policy, and our efforts over the course of a year have significantly helped improve client performance.

Because of the Internet's immense scale, test-beds are not a good approximation of network behavior. So a host of different diagnostic tools are used to evaluate the state of the network. *WhyHigh* uses active measurements such as TRACEROUTES and PINGS, in combination with datasets such as (a) BGP routing data, (b) end user RTT information, (c) Geo information of user prefixes and intermediate routers and (d) flow records in the network from a comprehensive measurement infrastructure in an attempt to pin down the cause for inflation to a particular customer ISP.

- *WhyHigh* uses diagnostic information described above to identify clients with poor latencies as ones which experience latencies much higher than that along the best path to the same region.
- Since clients in thousands of BGP prefixes can have inflated latencies, to help the administrators prioritize their efforts at troubleshooting, *WhyHigh* attempts to group together all prefixes likely affected by the same underlying cause.
- *WhyHigh* attempts to pinpoint the causes for the instances of latency inflation seen at each node to actionable items.

In Chapter 2, I examine state of the art in understanding BGP routing behaviour on the Internet. We describe how *WhyHigh* has helped ISP administrators detect path problems and identify their causes and improve overall routing performance.

This work was done in collaboration with Harsha Madhyastha, Sushant Jain, Prof. Arvind Krishnamurthy, Prof. Thomas Anderson and Prof. Jie Gao and has appeared as the following publication - 'Moving beyond end-to-end path information to optimize CDN performance', in the Proceedings of the 9th ACM SIGCOMM conference on Internet measurement, pages 190-201 where it won the best paper award [55].

1.2.2 GeoLocating Internet Hosts To Aid In Anomaly Detection

Accurate localization of Internet hosts and routers is an important aspect of diagnosing network performance issues. The *WhyHigh* tool uses location information of hosts and routers on the Internet to identify geographically circuitous paths, and accurate localization is an important aspect identifying such paths. It is also useful in a host of applications such as security applications, targeted advertising and location based services.

Several previous works [36, 51, 104] have addressed the problem of identifying coarse grained locations of Internet hosts based on network measurements. Although measurement based localization is not as accurate as those obtained using devices such as a GPS, they can be used to complement location information based on other sources and are especially useful in localizing intermediate hops for which accurate location hints are hard to find. In chapter Chapter 3 we analyze the state of art in measurement based localization and identify the cause of localization errors in existing algorithms.

In IP geolocation, there are a set of landmarks with known locations. They can issue probes to a given target host and collect round-trip delay data. These delay measurements are used as constraints in the problem of accurately identifying the target location. This is similar to localization problem in other settings such as Sensor networks. However IP geolocation has a number of unique challenges. Tools used for network measurement such as TRACEROUTE only have a uni-directional view of the network and hence cannot be used to detect links inflated due to asymmetric paths. Also techniques used to identify link latencies along the path can be inaccurate in the presence of MPLS tunnels or when some ISPs block traceroute packets from traversing their network. This results in a very incomplete picture with the possibility of measurement errors and missing information.

Existing algorithms try to use as many constraints as possible by using a large number of vantage points and extracting as many constraints as possible from the dataset to circumvent the measurement challenges. However we believe that choosing the right sub-set of constraints is also important in ensuring low localization errors. In this dissertation I use rigidity theory to identify the correct set of constraints and identify nodes with potentially low localization error.

I then propose a 2-tier localization algorithm. The first step is to identify nodes with potentially low localization errors as a set of secondary landmarks. These secondary landmarks can be used to improve the localization accuracy of many simple schemes which can be run in an online fashion. This work in collaboration with Prof. Jie Gao has been submitted for publication as ‘Improving Real-Time IP Geolocation of Internet Hosts’, to the 30th IEEE International Conference on Computer Communications.

1.2.3 Improving Routing In Wireless Mesh Networks Using Channel Aware Metrics

Wireless mesh networks are multi-hop wireless networks which are gaining popularity as last hop access technologies. These networks consist of wireless nodes, few of which act as gateways to the Internet. All the other nodes access the Internet through these gateways. The role of a routing protocol in these networks is to find paths that yield high throughput, low loss rates, and experience little variability. These networks are typically under a single administrative domain and most hosts are within a few hops of the gateway.

Here the main challenges are (a) designing accurate link and path metrics which help choose paths which can yield better sustained throughput. (b) evaluating such design decisions on a good evaluation platform and under varying channel conditions.

We have developed a *Channel Characteristics-Aware Routing Protocol* (CARP) that (1) uses per-packet transmission time to accurately estimate the residual capacity of a wireless link, (2) employs a bandwidth probability distribution model to better approximate a wireless path’s capacity profile, and (3) applies multi-path routing to exploit diversity among alternative paths and deliver more robust throughputs. CARP is one of the first algorithms to incorporate the effect of channel quality fluctuations in its routing metric. Together these mechanisms identify better paths even when the channel experiences high temporal fluctuation. In order to empirically quantify the protocol's behaviour and performance gains in real-world settings, we have evaluated different protocol mechanisms on a reconfigurable multi-hop wireless test-bed called MiNT-m.

In the Chapter 4, I examine the state of art in wireless mesh network protocols and describe the design of CARP and how it improves upon existing algorithms. I also describe the evaluation of CARP on MiNT-m. This work was done in collaboration with Ashish Raniwala and Prof. Tzi-cker Chiueh and has appeared as the following publications - 'Design of a Channel Characteristics-Aware Routing Protocol', that appeared in 30th IEEE International Conference on Computer Communications, pages 2441-2449, and 'An empirical comparison of throughput-maximizing wireless mesh routing protocols' that appeared in the proceedings of the 4th Annual International Conference on Wireless Internet Conference, article 40.

1.3 Dissertaton Outline

This dissertation is organized as follows. Chapter 2 describes routing in the Internet and *WhyHigh* tool which helps content providers evaluate the performance of their Content Distribution Networks. Chapter 3 examines causes of errors in existing localization algorithms and proposes a 2-tier localization algorithm to improve these errors. Chapter 4 describes routing in Wireless Mesh Networks and the CARP routing protocol which incorporates the effects of temporal fluctuations on path quality.

Chapter 2

Internet Routing

2.1 Overview

The Internet is a network of networks comprising of over 24000 separate administrative domains also known as Autonomous Systems (AS) which are identified by a unique integer called the AS number that serves as their identify. While some ASes are owned by organizations such as Universities, Companies and Content providers others are owned by Internet service providers (ISPs). In this dissertation we use the term ISP and AS interchangeably to refer to an administrative domain.

Routing in the Internet today operates at two logically distinct levels. Intradomain routing is used by ISPs to reach destinations inside their own networks. Interdomain routing, is used by ISPs to reach destinations in other ISPs networks. As part of interdomain routing, neighboring ISPs carry traffic for each other based on bilateral contracts. Based on these contracts we can classify Autonomous systems as stub ASes, multi-homed ASes and transit ASes. The stub ASes and multi-homed ASes connect to end-users or belong to content distributors. However they mostly do not forward traffic for other networks and hence form the edge of the Internet. The transit ASes primarily carry and forward traffic for others and charge for this service, and form the backbone of the Internet.

Since contracts between different ISPs are based on financial considerations they can significantly influence interdomain routing. ISPs tend to select locally optimal interdomain routing paths because the cost that an ISP incurs for carrying a packet depends on the path inside the ISP and because ISPs have limited knowledge about other ISPs networks. Thus, for each packet that an ISP controls, it selects a path that minimizes the local cost, largely disregarding the cost incurred by other ISPs. This can lead to inefficient and unstable routing paths. Paths can be inefficient because locally optimal routing decisions can be poor from a global standpoint [72, 96, 99] For instance, early-exit routing leads to longer than necessary paths [96]. Longer paths are undesirable because they consume more resources in the network and the performance of most applications is inversely correlated with path length. Paths can be unstable because the actions of ISPs influence each other. Since an ISP's path selection is based largely on local concerns, it can adversely impact another ISP, for instance, by causing congestion.

Typically most ASes pay the transit ASes to carry their traffic to different destinations. Hence, financial considerations dictate how much traffic should go along which path. Also, based on the type of financial agreement, the transit AS itself maybe more interested in ensuring maximum resource utilization. With the advent of the World Wide Web a significant amount of traffic flows between a content provider and a customer. A customers AS could directly connect to a content provider's AS bypassing transit ASes. Even in such a case the two ASes may have opposing interests. A content provider may be interested in ensuring that clients can connect to the content with low latency, while the ISP for the customer maybe interested in maximizing resource utilization on their networks by reducing traffic on it.

2.1.1 Routing In The Internet

As mentioned earlier the Internet operates at two logically distinct levels. Intra-domain routing determines how ISPs route packets to destinations within their network and protocols used for this are called Interior Gateway Protocols (IGP). Inter-domain routing in the Internet determines how ISPs route to destinations in

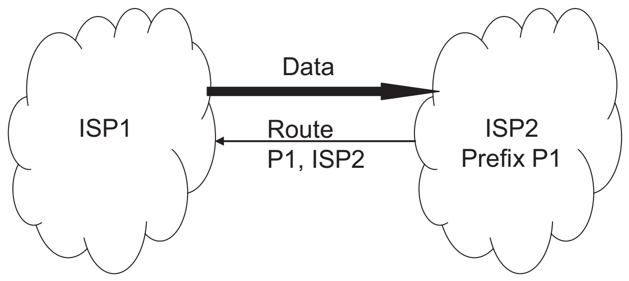

Figure 1: An ISP receives route advertisements from its neighbors. It then chooses the best path to a destination based on policy and uses that to send traffic.

other ISPs and this is handled by the Border Gateway protocol (BGP) [69, 88]. In this section we give a brief overview of BGP, as it is the main protocol used in the Internet. We also highlight limitations in BGP that make it hard for ISPs to control the quality of paths used to destinations.

BGP routers of different ASes talk to each other and exchange routing information. Each router advertises routes for a set of prefixes along with the AS level path to reach that prefix. BGP is a path vector protocol, which is similar to distance vector protocols, but as it includes the entire path to each destination in its advertisements it avoids some problems of distance vector protocols, such as count to infinity. Routing information flows in the direction opposite to the flow of data as shown in Figure 1. The routing information today includes the IP address prefix specifying the destination and the list of ISPs along the path, known as the AS-path, to the destination. It does not include information on the performance or cost of the path. ISPs are reluctant to share this information because they worry that any disclosed information might be abused by their competitors. For instance, if an ISP discloses the monetary cost of carrying traffic between pairs of cities, a competitor can potentially infer the ISPs profitable paths and use that information to undercut the ISPs profits by adding more capacity of its own along those paths. Also since routes are determined by what is advertised by an adjacent ISP they are not symmetric. Infact routing policies cause significant assymetry in routing.

When multiple paths to a destination are available to an upstream ISP, it uses local policies to select the path. The commercial relationships with neighboring

ISPs play a central role in these policies. Driven by monetary implications, ISPs usually prefer to send traffic through the lowest cost paths. Among equal cost paths, ISPs usually select paths based on AS-path length, assuming that it reflects end-to-end path quality. Among paths with the same AS-path length, ISPs use local optimization criterion (described below) as the basis for path selection. Early-exit routing, in which ISPs select the locally optimal interconnection while sending traffic to their neighbors, is an example of such a path selection policy. However, AS-path length is a poor indicator of end-to-end path quality in practice [99] as shown by previous work and our own analysis.

The original design of BGP [69] lacked support for optimizing network traffic, for instance, to balance network load, improve traffic performance, or reduce resource consumption. Over time, many ad hoc mechanisms have been added [88]. Some mechanisms that are commonly used today are multi-exit discriminators (MEDs), AS-path prepending and local preference (LPREF). MEDs are used between two ISPs that interconnect in multiple locations to influence how traffic enters the downstream ISP. The downstream ISP attaches ordinal preferences to routing messages which encode how it wishes to receive traffic to that destination. Whether the upstream honors MEDs is contractually determined by the two ISPs. When honoring them, the upstream ISP selects the interconnection that is most preferred by the downstream ISP. With AS-path prepending, the downstream ISP artificially increases the AS-path length in routing messages going out via certain interconnections by adding its AS number multiple times. This reduces the traffic on that interconnection if the upstream ISPs consider this path to be longer and are less likely to use it. LPREF allows an ISP to give higher preference to some of the paths among those advertised by the neighbor. The main difference between LPREF and MED is that LPREF is used to select outgoing paths within an ISP while MED attempts to control paths chosen by other ISPs.

BGP has proven to be a scalable, stable protocol for the Internet and provides some mechanisms for supporting routing policies. However this stability come at some expense of the extent to which an ISP can control the actual flow of data to a destination. BGP does not provide mechanisms to advertise the cost of different paths, and the granularity of the AS path is too coarse grained to determine performance. Also, the best policy path locally need not necessarily be the lowest latency or the highest throughput path and in this dissertation we show that this is often the case. Analyzing the performance of problems of particular paths in the Internet is especially difficult since policies of ASes are private and only the resulting paths are visible.

With the growth of the WWW, factors other than network utilization and reliability such as latency, response time, and bandwidth have gained importance. Content providers want their users to have good user experience which translates to lower latency, or higher throughput. Many content providers use Content Distribution Networks (CDN), where content is distributed among different geographically dispersed nodes, to provide good performance for their clients. However problems in the underlying routing paths can have significant affect on performance of CDNs. Also, since BGP does not have any performance metrics associated with the paths it is very hard to identify paths with problems. In this dissertation our goal is to identify the paths from a content provider to its clients which have poor performance by making use of other datasets available in this setting such as round trip times to different clients, netflow records within a content provider, number of connections from different ingress and egress points within a network etc. We present a tool called *WhyHigh* that allows administrators of CDNs to analyze and troubleshoot performance of paths to their clients. In particular we focus on improving client performance for the Google CDN. We also attempt to provide solutions within the framework of BGP to address them.

2.1.2 Routing In Content Distribution Networks

The use of content distribution networks (CDNs) has emerged as a popular technique to improve client performance in client-server applications. Instead of hosting all content on a server (or a cluster of servers) at a single location, content providers today replicate their content across a collection of geographically distributed servers or nodes. Different clients are redirected to different servers both for load balancing and to reduce client-perceived response times. The most common redirection method is based on latency, i.e., each client is redirected to the server to which it has the lowest latency in order to reduce the download time to fetch the hosted content. For example, several CDNs that run on PlanetLab use OASIS [33] to redirect clients to the node with least estimated latency.

Google's CDN similarly employs latency-based redirection. Common knowledge to optimize client latencies in such a setting is to establish more CDN nodes [43,100] so as to have a CDN node in the proximity of every client. However, we found that this does not suffice. For example, to reduce the latencies for clients in Japan when fetching content from Google, a new node in Japan was added to the Google CDN. While this decreased the minimum round trip times (RTTs) for clients in that region, the worst-case RTTs remained unchanged. This case showed that rather than investing in setting up new CDN nodes, we need to first understand whether the existing nodes yield the best performance possible.

In this chapter, we use measurements gathered at Google's nodes to study the effectiveness of latency-based redirection in optimizing CDN performance. We analyze RTTs measured to clients spanning approximately 170K prefixes spread across the world by monitoring the traffic Google exchanges with these clients. Our analysis of this data shows that even though latency-based redirection results in most clients being served by a geographically proximate node, the improvement in client-perceived RTTs falls short of what one would expect in return for the investment in setting up the CDN. We find two primary causes for poor latencies.

First, we find that geographically nearby clients often see widely different latencies even though they are served by the same CDN node. In our data, we find that clients in more than 20% of prefixes are served by paths on which RTTs are 50ms more than that observed to other clients in the same geographic region Figure 2. Such discrepancies in latencies are because some clients have circuitous routes to or back from the CDN node serving them. This leads us to the conclusion that latency-based redirection of clients alone is not sufficient to guarantee good performance. To reduce client latency a CDN administrator has to not only

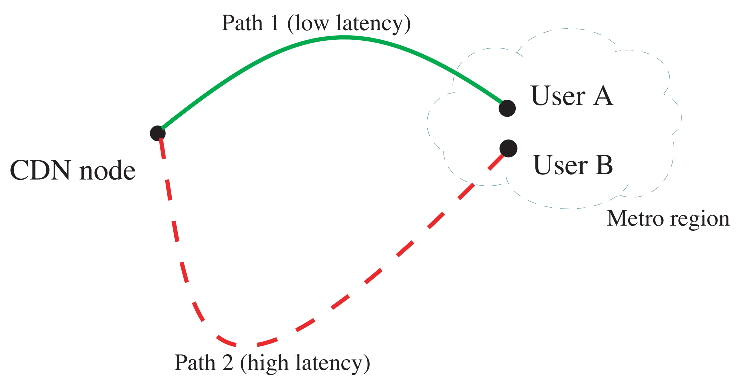

Figure 2: Although users A and B are located in the same metro region, they experience different latencies due to differing network paths.

distribute content geographically but also actively identify and troubleshoot sub-optimal paths in the network. To detect and diagnose these cases, we move beyond using end-to-end latency information and delve into the Internet's routing to optimize client performance. Second, we observe that connections to most clients are impacted by significant queueing delays. In this chapter, we characterize the extent of such queueing related overhead on client RTTs.

Given these problems, it is important that administrators be able to quantify the performance gains from the significant expense and effort that goes into deploying nodes in a CDN. Based on the analysis of our dataset and discussions with the network administrators, we have built a system called *WhyHigh* to aid administrators in this task.

WhyHigh first identifies clients with poor latencies as ones which experience latencies much higher than that along the best path to the same region. In a CDN that has several distributed nodes with each node serving clients in thousands of prefixes, identifying clients with poor performance is itself a significant challenge. Since clients in thousands of prefixes are determined to have inflated latencies, an administrator cannot investigate all of these simultaneously. To help the admins prioritize their efforts at troubleshooting, *WhyHigh* attempts to group together all prefixes likely affected by the same underlying cause. For example, we find that when a prefix suffers from inflated latencies, other prefixes served by the same AS path are also likely affected. *WhyHigh* also computes metrics that help compare different nodes in the CDN so that problems at the worst nodes can be investigated first to have higher impact.

WhyHigh then attempts to pinpoint the causes for the instances of latency inflation seen at each node. To infer these causes, *WhyHigh* combines several different data sources such as BGP tables from routers, mapping of routers to geographic locations, RTT logs for connections from clients, and traffic volume information. In addition, the system also performs active probes such as traceroutes and pings when necessary. Typical causes inferred by *WhyHigh* include lack of peering, routing misconfigurations, and side-effects of traffic engineering.

We have employed *WhyHigh* over the course of almost a year in troubleshooting latency inflation observed on paths between Google's nodes and clients. This has helped us identify and resolve several instances of inefficient routing, resulting in reduced latencies to a large number of prefixes. The metrics of CDN performance provided by *WhyHigh* have also allowed network administrators to monitor the results of their efforts by providing a global picture of how different nodes in the CDN are performing relative to each other.

In the rest of this chapter we look at the design and implementation of *WhyHigh* system and related work in trouble shooting path problems on the Internet. We then look at how localization of internet hosts helps in our goal.

2.2 Related Work

2.2.1 Content Distribution Networks

Content Distribution Networks (CDNs) have emerged as a commonly used platform to serve static content from the Web. A recent study stated that there are atleast 28 CDN providers as of 2007. Some popular examples include academic CDNs such as CoDeeN [102], and Coral [32], and commercial CDNs such as Akamai [5] and Limelight [68],

CoDeeN [102] is a CDN deployed on PlanetLab. Unlike commercial CDNs, accessing CoDeen requires explicitly setting up the proxy address in client's browser. The requests are redirected to a set of proxy servers which serve from their local cache. Cache misses are deterministically hashed and redirected to secondary proxy servers which only accept requests for a sub-set of URLs and cache data pertaining to these URLs thereby minimizing the load on the actual content servers. CoDeen proxies monitor their own health and the health of peers and redirect requests to less loaded peers.

Coral [32] is a decentralized peer to peer content distribution network. Coral CDN is open for any content provider to use, and is especially useful for small content providers with under provisioned servers to avoid sudden overloads. Unlike CoDeeN, content providers who are a part of Coral can publish their content with a special domain and take advantage of Coral's DNS redirection without the clients having to subscribe to a proxy. Also, content in Coral is cached using a specialized distributed hash table and servers can cache content from all providers are are not restricted to particular URLs. Since both Coral and Codeen are academic CDNs they don't maintain their own networks and many of the problems address in this chapter are not directly applicable.

Both Akamai and Limelight cache user content across a large number of servers distributed globally. They redirect clients to these servers via URL redirection in DNS servers which is transparent to users. A recent comparison study [43] between Akamai and Limelight highlighted the differences between these two CDNs. Akamai follows the design philosophy of entering deep into ISPs by deploying content distribution servers within POPs of other ISPs, while Limelight follows the philosophy of bringing ISPs to home by building large content distribution centers at few key points and peering with customer ISPs directly at these locations. The former case results in servers distributed worldwide which makes management more challenging. In the latter case the content provider is like an ISP and has to keep in mind the increased latency due to sub-optimal network paths.

In all these systems, different clients are redirected to different servers in an attempt to provide load balancing and to improve client performance. The common approach [33] employed to improve performance is to redirect a client to the server to which it has the least latency. The underlying assumption is that in the presence of a geographically distributed set of servers, latency-based redirection will direct the client to a nearby server to which the client has a low RTT. Our large scale measurement dataset from Google to clients all across the Internet shows that while redirecting based on latency does indeed lead to clients being served by nodes that are near-optimal with respect to geographical proximity, the expected improvement in latencies is far lesser than expected. Queueing of packets in routers and inefficient routing both are significant factors in the inflation of path latencies.

2.2.2 BGP And Network Path Inflation

Path inflation in the Internet has been studied [96] previously, and its causes have been traced predominantly to inter-domain routing policies. Previous work shows that upto 30% of Inter-domain network paths experience inflated latencies. They observed that path inflation is more common when ISPs follow hot-potato or early-exit policy rather than late exit policy. They also observed that Inter-domain paths were often sub-optimal because of AS path length not corresponding to shortest path. Extensions to BGP, the routing protocol used in the Internet, have been proposed [72] to provide ISPs the ability to cooperatively choose globally optimal routers without revealing their local routing policies. We find path inflation to be much more pronounced than observed previously. Even when considering RTTs from prefixes to geographically close nodes, we find that 30% of prefixes experience inflation greater than 50ms.

Overlay routing systems [7, 89] have attempted to circumvent inflated paths by routing through end-hosts that serve as relays. Instead, we focus on fixing instances of path inflation by identifying the underlying cause. Though the predominant causes of path inflation are known, we are the first to study how the cause for inflation on any given path can be identified. Our tool *WhyHigh* attempts to diagnose any specific path by pinpointing the AS along the path responsible for the inflation of path latency. This enables network administrators to discover whom they need to contact to improve the performance of their clients.

2.2.3 Measurement Tools

WhyHigh uses active measurements tools, such as TRACEROUTES and PINGS, if necessary. Traceroutes are a common network diagnostic tool, and have been used primarily to diagnose reachability issues and to pinpoint such problems to a particular AS, router, or link, e.g., PlanetSeer [109], Hubble [53], and [110]. We use traceroutes instead for performance diagnosis of paths, similar to iPlane [70] and Netdiff [74]. *WhyHigh* uses the geographic location of routers observed in the output of traceroute to determine if the path is circuitous. To do so, we use router DNS name to location mappings developed as part of the Rocketfuel project [97]. *WhyHigh* also attempts to group together prefixes affected by the same underlying cause, an approach used previously to diagnose BGP routing instabilities [31,105]. We also use TTL information to identify potential mismatch between forward and reverse paths. Traceroute latency to a prefix also provides an upper bound on minimum network latency to the prefix.

Many tools have been developed for measuring different path properties such as latency [76], bandwidth capacity [24,46], and loss rate [73]. The *WhyHigh* framework uses many different datasets to come up with a more coherent picture of the network and attempts to reduce the number of problems that administrators have to diagnose by grouping together paths that are affected by a common problem. Many of these tools can be easily incorporated into *WhyHigh* to provided richer diagnostic information.

2.3 Measurement Overview

In this section, we first provide a high-level overview of the architecture of Google's CDN deployment. Next, we present the goals of our work, while also clarifying what are non-goals. Finally, we describe the RTT dataset we gathered from Google's servers and present the preprocessing we performed on the data before using it for our analysis.

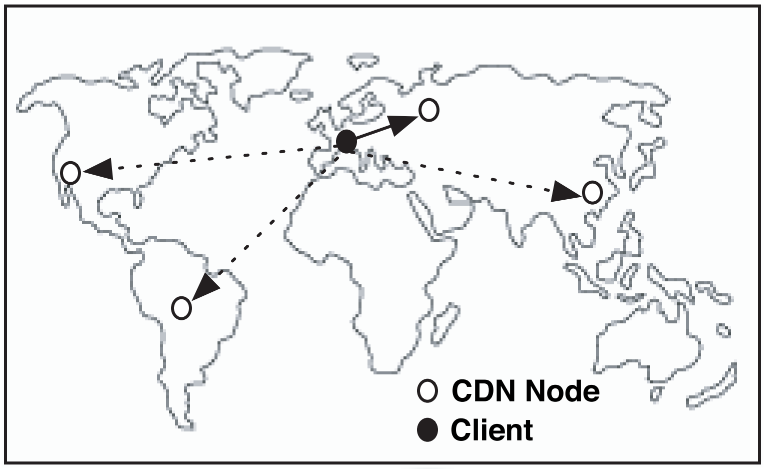

Figure 3: Architecture of Google CDN. Dotted lines indicate paths used primarily for measuring RTT. Bold line indicates path on which traffic is primarily served.

2.3.1 CDN Architecture

Figure 3 presents a simplified view of the architecture of the content distribution network operated by Google. Google's CDN comprises several nodes that are spread across the globe. To reduce the response times perceived by clients in fetching Google's content, the CDN aims to redirect each client to the node to which it has the least latency. For this, all clients in the same routable prefix are grouped together. Grouping of clients into prefixes is naturally performed as part of Internet routing, and the current set of routable prefixes are discovered from BGP updates received at Google.

To gather a map of latencies from CDN nodes to prefixes, every so often, a client is redirected to a random node and the RTT along the path between the client and the node is measured by passively monitoring the TCP transfer at the node. Since routing in the Internet is largely identical to all addresses within a prefix, the RTT measured to a client is taken to be representative of the client's prefix. Such RTT measurements gathered over time, and refreshed periodically, are used to determine the CDN node that is closest in terms of latency to each prefix. Thereafter, whenever a client attempts to fetch content hosted on the CDN, the client is redirected to the node determined to have the least latency to its prefix. This redirection however is based on the prefix corresponding to the IP address of the DNS nameserver that resolves the URL of the content on the client's behalf, which is typically co-located with the client.

2.3.2 Goals

In analyzing the performance offered by the CDN architecture described above, we have three primary goals.

- Understand the efficacy of latency-based redirection in enabling a CDN to deliver the best RTTs possible to its clients.
- Identify the broad categories of causes for poor RTTs experienced by clients.
- Implement a system to detect instances of poor RTTs and diagnose the root causes underlying them.

2.3.3 Non-goals

Apart from latency, TCP transfer times can also be dictated by path properties such as loss rate and bandwidth capacity. However, here we restrict our focus to optimizing end-to-end path latency, which is the dominant factor for short TCP transfers [18]. Further, our focus is on the component of end-to-end latency incurred through the network. In practice, response times perceived by clients can also depend on application-level overheads, e.g., delays incurred in rendering the content in a Web browser.

Latency-based redirection as described in the CDN architecture above can be rendered ineffective in a couple of scenarios. Clients which use a distant DNS nameserver may be redirected to a distant node, since the CDN will redirect the client to a node close to the nameserver. Redirection decisions made at the granularity of prefixes can be sub-optimal for prefixes in which clients are significantly geographically distributed. In this work, we focus on causes for poor client latencies even when the client's prefix is localized and the client is co-located with its nameserver.

2.3.4 Measurement Dataset

The dataset we use for most of analysis consists of RTT data measured by 15 nodes in Google's CDN to the 170K network prefixes they serve over the period of one day. The subset of 15 nodes used are spread across the globe and have coverage representative of the overall CDN. The median number of RTT samples per prefix is approximately 2000. For each TCP connection at a node, the node logs the time between the exchange of the SYN-ACK and SYN-ACK-ACK packets during connection setup; connections where TCP detects a packet loss and retransmits the SYN-ACK packet are ignored for the purposes of measurement. The RTT thus measured includes only the propagation and queueing delays on the forward and reverse paths traversed by the connection; transmission delay dependent on the client's access link bandwidth is not a factor because typically SYN-ACK and SYN-ACK-ACK packets are small. Though it is possible for the size of a SYN-ACK-ACK packet to be inflated by having a HTTP request piggy-backed on it, we found less than 1% of connections with such piggy-backed packets in a sample tcpdump.

Our analysis is based on RTT logs, BGP tables, and Netflow records obtained on 2 days—one in May 2008 and one in August 2008. We first use data from May to demonstrate the problems that clients face even after they have been redirected to the closest node in terms of latency. We later use data from August in Section 2.6 to illustrate the performance gains achieved through our efforts in troubleshooting poor client latencies.

2.3.4.1 Data Pre-Processing

We perform a couple stages of processing on our measurements before using them for analysis. First, we use BGP snapshots from different routers within the Google network to map every client to the routable prefix to which it belongs. Second, we tag prefixes with geographic information obtained from a commercial geolocation database. The location of each prefix was determined at the granularity of a region, which roughly corresponds to a province within a country. We applied three stages of processing to prune out prefixes with incorrect geographical information from our dataset.

First, we identify and eliminate any prefix such that the minimum RTT measured to it is impossible based on its estimated geographic location, i.e., RTT is much lesser than the time taken if a straight cable were laid between the node and the prefix. Prior work [51] on the geolocation of Internet hosts observed that the expected RTT through the Internet is that obtained when bits travel at $\frac{4}{9}^{th}$ the speed of light in vacuum. We refer to the RTT computed based on this assumption to be the *Geo-RTT* of a prefix. We eliminated all prefixes whose minimum RTT to any of the Google nodes was significantly lower than the Geo-RTT estimate to that node. This step prunes out 21K of the 173K prefixes in our dataset.

Second, we use the *"confidence"* information tagged along with data obtained from our geolocation database to discard information that is probably incorrect. Our geolocation product returns three attributes for every location—one each for country, region, and city. Each of these three attributes takes an integer value from 0 to 5, indicating the probability of the corresponding attribute being correct. The 0-to-5 scale equally divides up the confidence scale; a value of 5 indicates that the probability of the location being correct varies from 83% to 100%, a value of 4 for the range 67% to 83%, and so on. Since all of our analysis uses geolocation information at the granularity of region, we discard all prefixes whose location has a country confidence less than 5 and a region confidence less than 4. This prunes out a further 42K prefixes. In our significantly large sample of IP addresses with known locations used for validation, we find the 90th percentile error in location to be approximately 200 miles when restricting ourselves to locations associated with a country confidence of 5 and a region confidence greater than or equal to 4.

Third, some prefixes may span a large geographical region. For such prefixes, variation in RTT across clients could be because of their varying distance to the Google node serving that prefix. Since our focus here is on studying poor RTTs

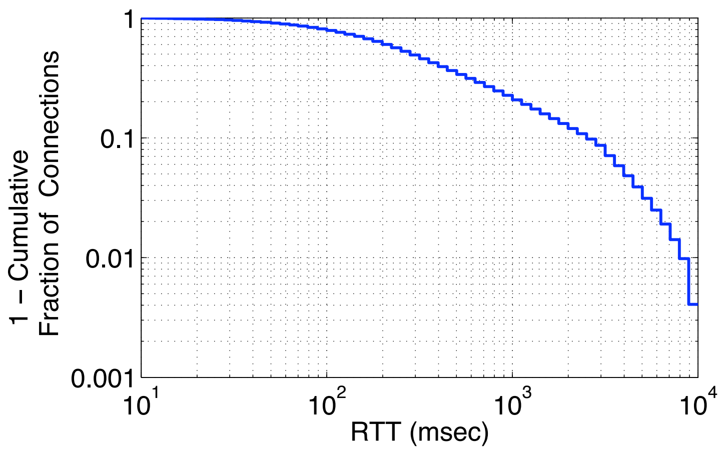

Figure 4: Distribution of RTTs measured across all connections. Note that the graph has a log-scale on both axes.

even when prefixes are geographically localized, we eliminate these distributed prefixes from consideration. To identify such prefixes, we compute the width of each prefix in the dataset by choosing a few hundred IP addresses at random from the prefix, determining the locations for these addresses, and computing the maximum distance between pairs of chosen addresses. We discard all prefixes with a width greater than 100 miles. This eliminates an additional 5K prefixes.

2.4 Understanding Path Latency

In this section, we analyze our dataset to understand the RTTs seen by clients of Google's CDN infrastructure. First, we study the efficacy of latency-based redirection. Then, we investigate each potential cause for poor performance.

The motivation for this work lies in the poor latencies seen by clients. Figure 4 plots the distribution of RTTs measured across all connections in our dataset. Even though each client is served by the CDN node measured to have the lowest latency to the client's prefix, RTTs greater than 400ms are measured on 40% of the connections served. We would expect to see significantly lower RTTs given the spread of Google's CDN; roughly 75% of prefixes have a node within 1000 miles (Geo-RTT less than 20ms) and roughly 50% of prefixes have a node within 500 miles (Geo-RTT less than 10ms). Moreover, 400ms is more than the typical RTT obtained on a path all the way around the globe, e.g., the RTT between PlanetLab nodes on the US west coast and in India is roughly 300ms. This shows that in spite of replicating content across several CDN nodes to ensure that every client has a server nearby, which requires significant investment in infrastructure, the latencies experienced by clients are poor.

To understand the cause for these high RTTs, we break down each measured connection RTT into its components. RTTs measured at the TCP layer have three main components—transmission delay (time to put a packet on to the wire), propagation delay (time spent from one end of the wire to the other end), and queueing delay (time spent by a packet waiting to be forwarded). Since our measurements are based on control packets typically of size 50 bytes, transmission delay is less than 1ms even on a dialup link. Therefore, a high RTT is the result of inflation in either propagation delay or queueing delay. Inflation in propagation delay could be the outcome of two factors. On the one hand, it could be the case that clients are far away from the node to which they have the lowest latency; the node with lowest latency from a client need not necessarily be the same as the node geographically closest to the client. In such a case, even though a client is redirected to the node with lowest latency, its latency to this node can be high because of the distance between it and the node. On the other hand, even if clients are being redirected to nearby nodes, a client could have high RTT to the nearby node to which it is redirected. Beyond propagation delay, high RTTs could also be the result of packets getting queued up somewhere along the path between the clients and the nodes serving them. We next examine each of these potential causes for poor latencies in detail.

2.4.1 Effectiveness Of Client Redirection

We first evaluate whether the cause for high RTTs is because clients are being redirected to nodes distant from them. For every prefix, we identify the main node

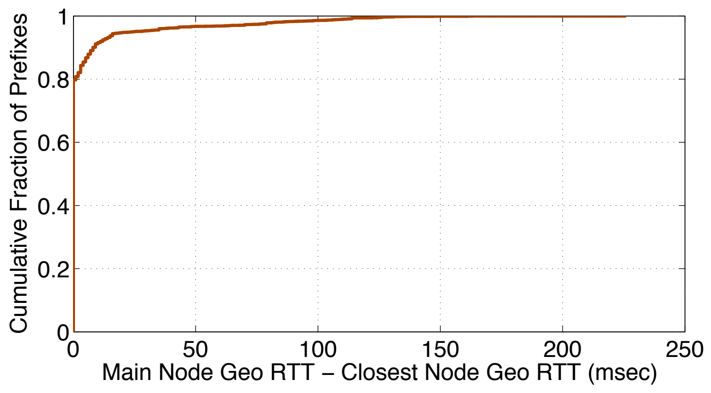

Figure 5: CDF across prefixes of the difference between Geo-RTTs from a prefix to the main node serving it and to its closest node. Most prefixes are served by geographically nearby nodes.

serving it, i.e., the node from which most of the connections to this prefix were observed. We also identify for every prefix, the node geographically closest to it. Based on the CDN architecture described previously, the main node for each prefix should be the node to which it has the lowest RTT. However, there are a few cases where this property is violated. Clients can be redirected elsewhere if the lowest RTT node is under high load, the nameserver used by the client is not co-located with it, or network conditions change between when the RTT to the client's prefix was measured and when that information is used to make redirection decisions. Since our goal is to evaluate the performance of the CDN when clients are served by the node to which they have the least RTT, we drop all prefixes from our dataset where that property is not true. This prunes out an additional 14K prefixes.

Figure 5 plots the difference between the Geo-RTT, as previously defined, from a prefix to its main node and the Geo-RTT from that prefix to its geographically closest node. Clients in 80% of prefixes are served by their geographically closest node. Further, 92% of the prefixes are redirected to a node that is within 10ms of Geo-RTT from their closest node, which corresponds to a distance of 400 miles. This shows that latency-based redirection does result in most prefixes being served by nearby nodes, albeit we show later in Section 2.5 that this is not true for new nodes added to the CDN.

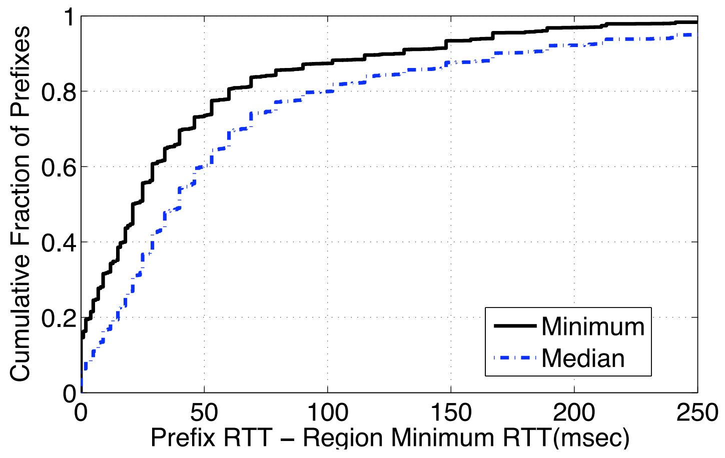

Figure 6: Distribution of the inflation in latency observed in the minimum and median RTT to a prefix as compared to the minimum RTT to any prefix in the same region. Inflation seen in a prefix's minimum RTT is predominantly due to routing inefficiencies, and additional inflation seen in the median is due to packet queueing.

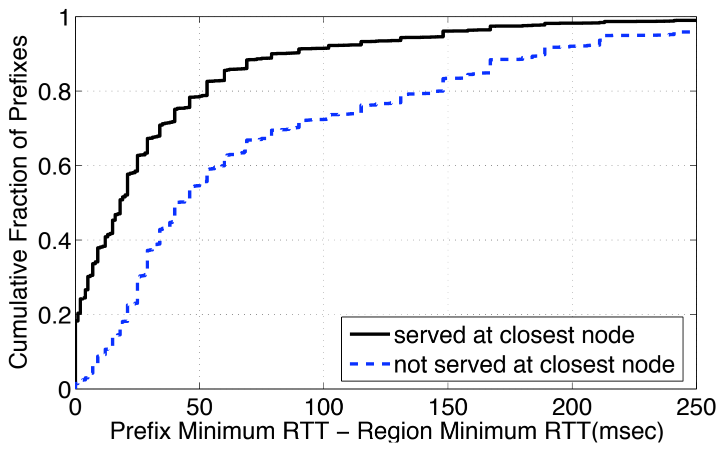

Figure 7: Latency inflation w.r.t other prefixes in the region for (i) prefixes redirected to the closest node, and (ii) prefixes served by nodes farther away.

2.4.2 Characterizing Latency Inflation

We next investigate large propagation delay to nearby nodes as the potential cause for the high measured RTTs. For this, we compare the following two values. First, we compute the minimum RTT seen across all connections originating at a prefix. Second, we determine the minimum RTT measured across all prefixes in the same region. Both of these are computed across all measurements made at the prefix's main node. Given the large number of RTT samples in each prefix, the minimum RTT to a prefix is unlikely to have a queueing delay component. The solid line in Figure 6 makes this comparison by plotting the difference between these two values. We see that more than 20% of prefixes experience minimum RTTs that are more than 50ms greater than the minimum RTT measured to other prefixes in the same region. This problem is even worse when considering prefixes served by new nodes, i.e., nodes active for less than 1 year; 45% of prefixes served at new nodes have minimum RTTs that are 50ms greater than other RTTs measured to the region (not shown in figure). This shows that although most prefixes are served by a node that is geographically close, and there exist good paths to the regions of these prefixes, routing inefficiencies result in inflated RTTs. Though our methodology will miss instances of inflation when all prefixes in a region are affected, we find that cases where the best RTT to a region is poor are rare.

These numbers show that the problem of inflated latencies due to inefficient routing is significantly worse than previously considered. Prior work [96] that looked at inflation between pairs of cities had found less than 5% of paths to have an inflation greater than 25ms. Our analysis of a much more extensive RTT dataset measured to several tens of thousands of prefixes across the Internet shows that more than 20% of paths have an inflation greater than 50ms, even though most paths are between clients and nearby nodes.

To further examine the inflation observed in the minimum RTT to a prefix, we partition prefixes in our dataset into two sets—1) prefixes that are redirected to the node geographically closest to them, and 2) all other prefixes (which are not redirected to their geographically closest node). Figure 5 shows that 80% of prefixes belong to the first partition and 20% belong to the second.

A prefix would be redirected to a node other than the geographically closest one only when there exists another node with lower latency to the prefix. Therefore, one would expect prefixes in the first set to have lower inflation, on average. To examine this hypothesis, Figure 7 plots the inflation observed in the minimum RTTs for prefixes in both sets. We see that even prefixes redirected to their closest node

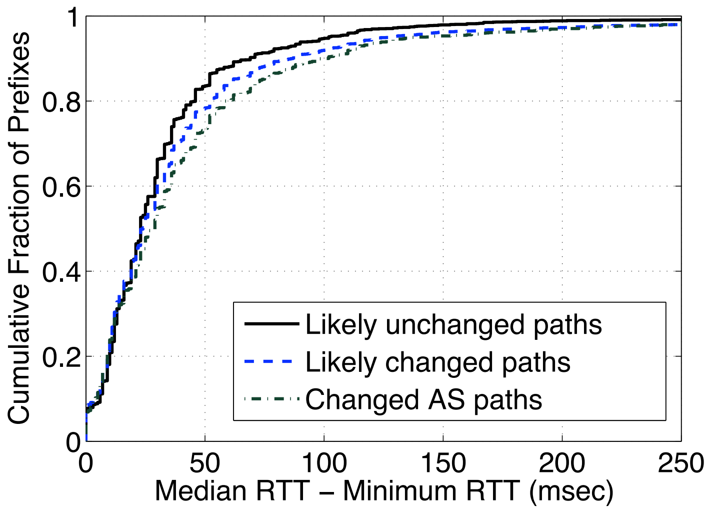

Figure 8: Comparing latency inflation between sets of prefixes partitioned based on whether the path to them changes across consecutive days.

suffer from significant latency inflation—20% of prefixes have inflation greater than 50ms—and this is even more pronounced for prefixes redirected to farther servers. When considering only the prefixes served by new nodes (not shown in figure), this problem is even worse—in either set, 45% of prefixes suffer greater than 50ms inflation.

2.4.3 Characterizing Queueing Delays

Next, we analyze the other potential cause for high RTTs—packet queueing. Going back to Figure 6, we plot the inflation observed in the median RTT measured to a prefix compared to the minimum RTT to its region. We see that the additional latency overhead observed is significant; 40% of the prefixes incur an inflation greater than 50ms when considering the median. The variance in RTTs across different clients in a prefix because of geographic diversity is limited by the fact that our dataset only contains prefixes whose geographic width is at most 100 miles, which accounts for less than 5ms of RTT in the worst case. Also, variance in access link bandwidths across clients in a prefix cannot explain a difference of tens

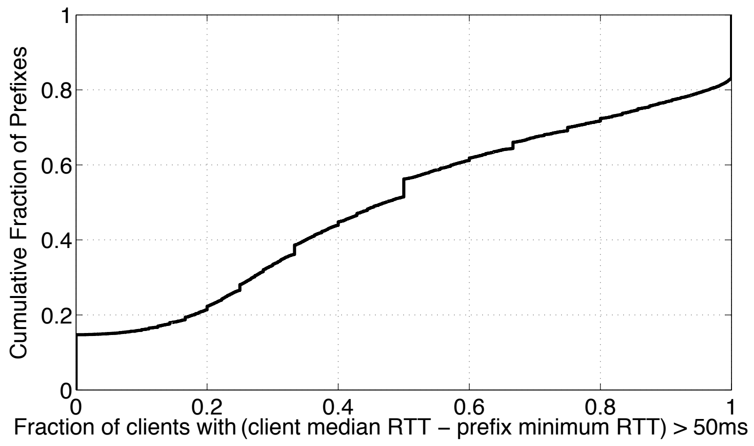

Figure 9: Fraction of clients affected by significant queueing delays in inflated prefixes. In most inflated prefixes, most clients experience high latency overhead owing to queueing.

of milliseconds as our measurements are made over TCP control packets whose sizes are typically in the order of 50 bytes.

Therefore, the significant variance within RTTs measured across connections to the same prefix can be due to two reasons—1) the route between the prefix and its main node could vary over the course of a day, which is the duration of our dataset, or 2) packets could be getting queued along the route. To investigate the former, we ranked the prefixes based on the overhead measured in the median RTT to that prefix as compared to the minimum to the prefix. We considered the top 10K prefixes based on this ranking and issued traceroutes to them from their main nodes over the course of a day. We then re-issued these traceroutes the next day and partitioned the prefixes based on whether the routes measured were identical across the two days. We determine the two routes to a prefix across successive days to be identical if either of the following two conditions are met. First, we compared the two routes at the granularity of router interfaces as measured by traceroute. If deemed non-identical, second, we mapped the routes to the granularity of Points-of-Presence (PoPs) using data from iPlane [70] and compared the PoP-level routes. 4K prefixes had routes that satisfied either of these two matching criteria, and we consider routes to these prefixes to be likely unchanged across the two days. The remaining 6K prefixes had routes that likely changed across days.

Figure 8 plots the overhead seen in comparing the median and minimum RTTs to a prefix for prefixes in these two partitions. The distribution of latency overhead is pretty much identical irrespective of the likelihood of the route to the prefix being stationary.

Prior work [71, 79, 111] has found more than two-thirds of routes to be identical across consecutive days. Therefore, some of the prefixes for which we could not match routes at the granularity of either router-interfaces or PoPs, could also be stationary. The incompleteness of our data mapping IP addresses to PoPs could limit us from identifying route stationarity in some cases where the paths are indeed identical. Therefore, to select a subset of prefixes to which routes have certainly changed across days, we map traceroutes to AS-level paths and compare them. Again, we use data from iPlane [70] to map IP addresses to ASes. 1700 prefixes of the 10K considered for this experiment are determined to have different AS paths. Figure 8 shows that the distribution of inflation for this subset of certainly unstationary prefixes too is similar to that for the subset previously determined to be likely stationary.

In this experiment, our determination of whether routing to a prefix is stationary was based only on the routes we measured from the CDN nodes to these prefixes and measured once per day. Therefore, we might have missed out on changes in routing on the path back from the prefix to the node or at finer timescales. However, the fact that accounting for even the limited degree of stationarity that we detect does not make a difference implies that queueing of packets, and not flux in routing, is the predominant cause for the huge variance observed in RTTs within a prefix. While our data does not enable us to determine whether packets are being queued in routers, in access links, or within the operating system of clients, large queueing delays have previously been measured [23] on the access links of end-hosts.

To determine if packet queueing is restricted to a few clients within each prefix and not to others, we conducted the following experiment. We considered the top 25K prefixes in the ranking computed above based on the difference between the median and minimum RTTs to the prefix. In each of these prefixes, we computed for every client the difference between the median RTT measured to the client and the minimum RTT to the prefix, i.e., the median RTT inflation experienced by the client due to queueing. Figure 9 plots the fraction of clients in each prefix for which this difference is greater than 50ms. We see that in roughly half of the prefixes, more than half the clients have an overhead greater than 50ms in the median RTT as compared to the minimum to the same prefix. This shows that latency overheads caused by packet queueing are widespread, not restricted to a few clients within each prefix.

2.4.4 Summary

The results in this section demonstrate that although redirection of clients is essential to reduce client latency, that alone does not suffice. Redirection based on end-to-end RTTs results in most clients being served from a geographically nearby node. However, two factors result in significant latency inflation. First, a significant fraction of prefixes have inefficient routes to their nearby nodes, a problem which is even more pronounced when considering prefixes served by CDN nodes active for less than a year. Second, clients in most prefixes incur significant latency overheads due to queueing of packets. Isolating the cause or fixing the overheads caused by packet queueing is beyond the scope of this work. We focus on diagnosing and fixing routing inefficiencies that lead to poor client RTTs.

2.5 WhyHigh: A System For Diagnosing Latency Inflation

In this section, we describe the design and implementation of *WhyHigh*, a system that we have built to detect and diagnose cases of inefficient routing from nodes in Google's CDN to clients. We follow up in the next section with a few representative cases of path inflation diagnosed by *WhyHigh* and provide an overview of the progress we have made in our efforts towards fixing the problems presented in Section 2.4.

In building *WhyHigh* to diagnose poor RTTs experienced by clients, we follow a three-staged approach— 1) *identify* prefixes affected by inefficient routing, 2) *prioritize* these prefixes for investigation by administrators, and 3) *diagnose* causes for the identified problems. We now describe the techniques used by *WhyHigh* in each of these subcomponents in detail.

2.5.1 Identifying Inflated Prefixes

At each CDN node, we measure the RTT over every connection. We then identify prefixes with inflated RTTs, which we refer to as *inflated prefixes*. To identify inflated prefixes at a node, we compare the minimum RTT measured at the node across all connections to the prefix with the minimum RTT measured at the same node across all connections to clients within the prefix's region. We declare a prefix to be inflated if this difference is greater than 50ms. As we argued previously in Section 2.4, inflation in latency observed in the minimum RTT to a prefix is because of inefficient routing. Thus, we use all inflated prefixes identified using this process as candidates for further troubleshooting.

2.5.2 Identifying Causes Of Latency Inflation

The latency to a prefix can be inflated because of a problem either on the forward path from the CDN node to the prefix or on the reverse path back. To isolate the cause for inflation on either the forward path to or reverse path from an inflated prefix, *WhyHigh* uses data from BGP. Snapshots of the BGP routing table provide information on the AS path being used to route packets to all prefixes. A log of all the BGP updates tells us the other alternative paths available to each prefix. In addition, to obtain routing information at a finer granularity than an AS and to obtain visibility into the reverse path back from prefixes, *WhyHigh* also performs two kinds of active measurements—a traceroute [1] from the node to a destination in the prefix, and pings to intermediate routers seen on the traceroute. *WhyHigh* then uses these datasets to isolate whether the cause of inflation for an inflated prefix is on the forward or the reverse path.

To identify circuitousness along the forward path from a CDN node to a prefix, *WhyHigh* uses the sequence of locations traversed along the traceroute to the prefix. To identify circuitousness along the reverse path, *WhyHigh* uses three different techniques. First, it uses a significant RTT increase on a single hop of the traceroute to be an indicator of reverse path inflation. A packet that traverses any single link usually experiences a one-way latency of at most 40ms. We observed one-way link latencies to be less than 40ms on various long distance links that span thousands of miles (e.g., from coast to coast in the US, trans-Pacific, trans-Atlantic, and across Asia). Hence, an increase of 100ms or more in the RTT measured on successive hops of a traceroute likely indicates that the reverse path back from the latter hop is asymmetric. Second, the return TTL on the response from a probed interface provides an estimate of the length of the reverse path back from the interface. Therefore, a significant drop in the return TTL on pings to two successive interfaces on the traceroute implies that the reverse path back from at least one of the hops is asymmetric; comparison with forward TTLs tells us which one.

However, changes in RTT and return TTL across hops only provide hints about inflation on the reverse path. These do not suffice to identify the exact path taken by packets from the client to the node. To gain partial visibility into the reverse path, *WhyHigh* uses flow records gathered at border routers in Google's network; flow records are summaries of traffic forwarded by the router. Whenever records for flows to a client are available at a specific router, we can confirm that the reverse paths from that client pass through this router. We use this to compute the entry point of client traffic into the Google network. Therefore, if the path from a prefix to a node enters the Google network at a distant location from the node, we can infer circuitousness along the reverse path. In some cases, these techniques do not

[1]Traceroutes are performed with limited max TTL to ensure no probes reach the destination.

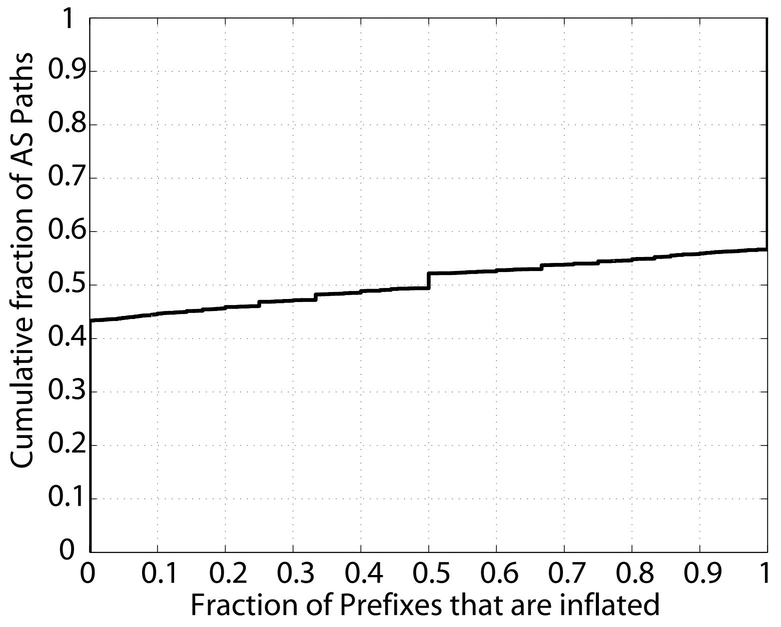

Figure 10: Fraction of prefixes served by an AS path that have inflated latencies. For most paths, either all prefixes have inflated latency or none of them do.

suffice for enabling *WhyHigh* to characterize the cause for inflation to a prefix to be either along the forward or reverse path. We, however, show later in Section 2.6 that such cases account for a small fraction of inflated prefixes.

2.5.3 Identifying Path Inflation Granularity

As seen previously in Figure 6, more than 20% of all prefixes are inflated, i.e., tens of thousands of prefixes are inflated. From the perspective of the CDN administrators, troubleshooting all of these prefixes individually is an intractable task. To make the administrator's task easier, *WhyHigh* leverages information from BGP paths to group prefixes potentially affected by a common underlying cause. For example, if a routing misconfiguration on a particular router is responsible for inflation on the path to a prefix, similar inflation is likely to be seen for other prefixes that have to traverse the same router.

Hence, *WhyHigh* merges prefixes into the largest set possible where all the prefixes in the set suffer from inflated latencies. The various granularities at which it attempts to group prefixes are: (i) prefixes sharing the same PoP-level path measured by traceroute, (ii) prefixes sharing the same AS path and the same exit and entry PoPs out of and into Google's network, (iii) prefixes sharing the same AS path, and (iv) prefixes belonging to the same AS, in that order. Other means of grouping prefixes could also prove to be useful, e.g., all prefixes served by a common intermediate shared AS or AS path. However, we are still exploring the incorporation of these other techniques for grouping prefixes, while ensuring that the fraction of inflated prefixes in a group remains high.

Figure 10 demonstrates the utility of grouping together prefixes by using data from one CDN node. For each AS path along which traffic is forwarded from this node, the figure plots the fraction of prefixes served by that AS path that are inflated. As seen in the figure, path inflation typically affects either all prefixes served by an AS path or none of the prefixes served by it. Hence, most instances of path inflation can be addressed at a higher granularity than prefixes, such as at the level of an AS path or an AS. This significantly reduces the problem space to be tackled by an administrator and lets her focus first on AS paths or ASes that account for a large number of inflated prefixes.

Grouping prefixes as above also helps *WhyHigh* isolate causes for inflation by providing information on what are possibly *not* the causes. For example, from a particular node, if the paths to all prefixes served by the AS path *path1* are inflated, and none of the paths to prefixes served by the AS path *path2* are, then one of the ASes on *path1* that is not on *path2* is likely to be responsible for the inflation.

2.5.4 Ranking Nodes

Another means by which *WhyHigh* helps administrators prioritize problems is by ranking the nodes in the CDN. At nodes afflicted by more widespread latency inflation, a single problem is typically the cause for inflation to several prefixes. Hence, following through with a single troubleshooting effort can have significant impact.

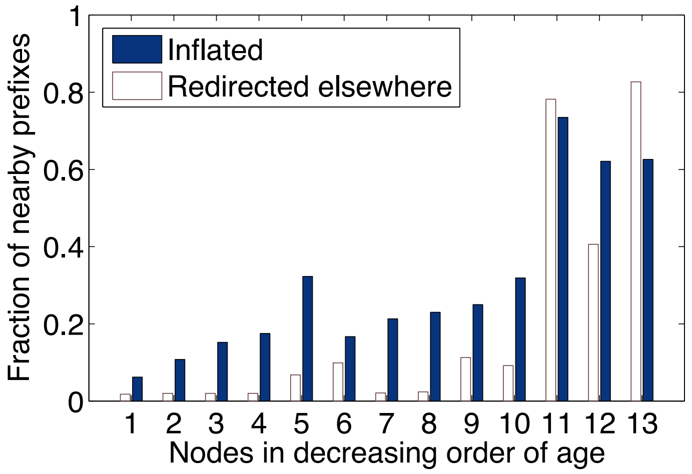

Figure 11: Of the prefixes geographically closest to a node, fraction that are inflated and fraction that are served elsewhere, i.e., not at the prefix's closest node.

WhyHigh can rank order nodes in the CDN based on several metrics. A couple of example metrics quantifying a node's performance are—1) the fraction of nearby prefixes (which ought to be served at the node) that have inflated latencies, and 2) the fraction of nearby prefixes that are served elsewhere. A high value for either metric indicates a poorly performing node. Figure 11 plots both metrics for the subset of 13 nodes that we consider in Google's CDN. The nodes are ordered in decreasing order of their age from left to right. This graph shows that new nodes are much more likely to have more widespread latency inflation issues, and unlike previously seen in Figure 5, a large fraction of prefixes near new nodes get redirected elsewhere. Therefore, the resources of an administrator are better spent troubleshooting problems diagnosed by *WhyHigh* at these nodes.

2.5.5 Identifying Root Causes Of Inflation

After having gathered all relevant information for every inflated prefix, *WhyHigh* attempts to pinpoint the set of causes for these problems. An administrator then goes over this list of causes, prioritizing them based on their impact, to verify them and to follow through on the proposed corrective actions.

The plausible causes that *WhyHigh* attributes to instances of latency inflation can largely be binned into four classes.

- **Lack of peering** [2]: When all available AS paths from a node to an inflated prefix are long even though the prefix's AS has presence in the same region as the node, having Google peer with the prefix's AS could fix the problem.
- **Limited bandwidth capacity**: In cases where the inflated prefix is in an AS that peers with Google and yet a circuitous path is being used to forward traffic from the node to the prefix, limited bandwidth capacity on the peering link between Google and the prefix's AS is the likely cause. To fix the latency inflation, more bandwidth needs to be provisioned to carry traffic from the node to all prefixes in the AS.
- **Routing misconfiguration**: *WhyHigh* attributes an instance of latency inflation to an error in routing configuration when the inflated prefix is in an AS which peers with Google, and though the direct path to the peering AS is being used, the reverse path back from the prefix is circuitous. The peering AS then needs to be contacted to correct its routing configuration.
- **Traffic engineering**: When an inflated prefix is not in an AS with whom Google peers, if a long AS path is being used from the node to the prefix even though alternate shorter paths exist, the latency inflation is likely due to traffic engineering. The network administrators who put in place the traffic engineering policies may not be aware of this side-effect, and hence, alerting them about the resulting latency inflation may help find a fix.

2.5.6 System Architecture

Figure 12 summarizes the steps involved in the *WhyHigh* pipeline. *WhyHigh* first gathers relevant logs from different data sources. It then associates each routable prefix with a host of diagnostic information based on data from these logs. This information includes 1) the RTT distribution observed by clients within a prefix, 2) the AS path to this prefix observed from all CDN nodes, 3) the geographic

[2]**We use the term *peering* loosely to refer to any kind of connection between two ASes.**

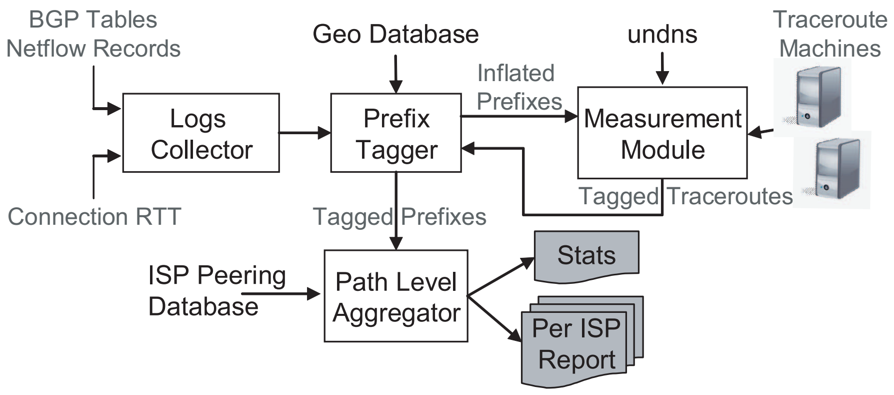

Figure 12: System architecture of *WhyHigh*.

location of the prefix, and 4) the set of border routers within Google's network that have observed traffic to or from this prefix.

Once all prefixes have been tagged with relevant information, the *WhyHigh* system identifies prefixes having abnormally high latencies and performs active measurements from different nodes to these prefixes. This ensures that we only actively probe potential problem cases. The system then tags prefixes as having circuitous forward or reverse paths based on techniques described above.

WhyHigh outputs two kinds of information. First, it generates general statistics which help administrators monitor the performance of different nodes in the CDN. Second, it generates a per AS report as shown in Figure 13, which identifies inflated latency AS paths to prefixes in the AS and identifies the potential causes of inflation. In addition, to help in troubleshooting, *WhyHigh* also identifies good paths to the AS for comparison.

2.6 Results

In this section, we present some results from our experience with running the *WhyHigh* system over the period of several months. First, we present a few representative instances of inflated latencies that were diagnosed using *WhyHigh*. We

AS Info	
AS Name, Country	Peering Locations
Statistics:	(a) Number of inflated prefixes
	(b) Number of affected connections
	(c) Expected improvement
Un-affected Paths	
Path:	(AS path1, Entry PoP1, Exit PoP1)
Prefixes:	P1, P2, ...
Affected Paths	
Path:	(AS path2, Entry PoP2, Exit PoP2)
Prefixes:	P3, P4, ...
Diagnostic Info	(a) Forward path inflation cases
	(b) Reverse path inflation cases
	(c) Culprit ASes
	(d) Sample traceroutes

Figure 13: Format of report produced at every Google node by *WhyHigh* for each AS to which inflated latency paths are observed.

then present some results summarizing *WhyHigh*'s classification of detected problems and the improvement in client latencies as a result of following through on *WhyHigh*'s diagnosis.

2.6.1 Illustrative Example Cases

We employed *WhyHigh* to diagnose the causes for several instances of path inflation. Here, we present a few illustrative examples that are representative of the different types of causes we encountered for path inflation.

Case 1: No peering, and all AS paths are long

WhyHigh reported that a node in southeast Asia consistently observed RTTs of more than 300ms to prefixes in PhilISP1 [3], an ISP in the Philippines. This was 200ms greater than the minimum RTT measured for that region. Given the distance between this node and the Philippines, such high RTTs are unexpected, even though

[3]Some of the AS names have been anonymized.

ANC–Sprint–PhilISP1
ANC–Tiscalli–PhilISP1
ANC–Level3–NTT–PhilISP1
ANC–Level3–Teleglobe–PhilISP1
ANC–Level3–MCI Verizon–PhilISP1
Flag–PhilISP1
BTN–PhilISP1

Figure 14: AS paths used in troubleshooting Case 1. All paths go through the US en route to the destination in Philippines.

Google does not directly connect with PhilISP1.

We examined the AS paths received at Google for PhilISP1's prefixes (shown in Figure 14). PhilISP1 splits its address space into several more specific prefixes and distributes these across its providers; we believe PhilISP1 is doing so to load-balance the traffic it receives across all of its peerings. PhilISP1's neighboring ASes on these AS paths have links with Google's providers only in the US. So, all the AS paths received by Google for reaching PhilISP1 traverse the US. The resultant circuitous routes on both the forward and reverse paths lead to RTTs greater than 300ms from the node in SE Asia to destinations in PhilISP1. Based on this information, *WhyHigh* diagnosed the cause for this case as the lack of peering between Google and PhilISP1.

Case 2: No peering, and shorter path on less specific prefix

A node in India measured RTTs above 400ms to destinations in IndISP1, an ISP in India. Though Google and IndISP1 do not directly peer, such high RTTs within a country are unreasonable.

WhyHigh revealed the following. As shown in Figure 15(a) [4], the traceroute from the Indian node to a destination in IndISP1 shows the forward path going through ApacISP, an ISP in the Asia-Pacific region, via Japan. ApacISP further

[4]Hostnames have been anonymized in all traceroutes.

```
1.  1.1.1.1                        0.116 ms
2.  1.1.1.2                        0.381 ms
3.  japan1.nrt.google.com          145.280 ms
    (nrt → Narita, Japan)
4.  exchangepoint.jp               146.327 ms
5.  router.lax.apacisp.com         262.749 ms
    (lax → Los Angeles, USA)
6.  address1                       509.779 ms
```

(a)

Prefix Length	Peering Point	AS path
/16	India	IndISP2–IndISP1
/18	SE Asia	ApacISP–Reliance–IndISP1
/18	Japan	ApacISP–Reliance–IndISP1

(b)

Figure 15: Data used in troubleshooting Case 2: (a) Extract of traceroute, and (b) AS paths received by Google.

worsens the inflation by forwarding the traffic through Los Angeles. This circuitous route causes path inflation for this case. *WhyHigh* reported better alternate paths for IndISP1's prefixes. In fact, as seen in Figure 15(b), we found that other than the path through ApacISP that was being used to carry the traffic, an AS path via IndISP2 was also received by Google in India. Since IndISP2 is an Indian ISP too, it surely connects with IndISP1 at some location in India. So, if this alternate AS path had been chosen instead to forward the traffic, we would observe much smaller latencies to end-hosts in IndISP1. However, this path is not an option since the prefixes associated with the AS paths through ApacISP are more specific than that associated with the AS path via IndISP2. On further investigation, this turned out to be due to the lack of capacity on the peering between IndISP2 and IndISP1.

Case 3: Peering, but longer paths on more specific prefixes

WhyHigh reported that connections served from the node in southeast Asia also experienced RTTs over 300ms to prefixes in PhilISP2, another ISP in the Phillipines. Unlike the previous two cases, in this instance RTTs are inflated even though Google connects with PhilISP2 in the region.

We analyzed the diagnosis information provided by *WhyHigh* for this case. Google receives a direct AS path to PhilISP2 for the prefixes that encompass all of PhilISP2's globally advertised address space. However, Google does not use this AS path for any traffic since PhilISP2 splits its address space into several smaller prefixes and announces these more specific prefixes to ASes other than Google. All of the AS paths through these other neighbors of PhilISP2 go through some AS in the US, such as AT&T or Level3. A router determines the route on which to forward traffic by performing a longest prefix match of the destination against the list of prefixes for which it received routes. Hence, the direct path from Google to PhilISP2 is never used. Routers forward all traffic to PhilISP2's prefixes over these AS paths that go through the US, resulting in extremely high end-to-end latencies.

Case 4: Peering, but inflated reverse path

In our final representative example, we look at a case reported by *WhyHigh* for JapanISP, an ISP in Japan that directly connects with Google. The RTT from the node in Japan to a set of prefixes in Japan hosted by JapanISPwas over 100ms. Again, this problem is worse than it first appears since Google connects with JapanISP in Japan.

To diagnose this problem, we looked at the traceroute, shown in Figure 16(a), from the node in Japan to an address in one of the problematic prefixes. The RTT increased from around 1ms to over 100ms across the hops at either end of the peering link between Google and JapanISP, indicating inflation on the reverse path back from JapanISP. An increase of 4 hops in the reverse path length estimated by pings to either end of the peering link further substantiates the reverse path asymmetry (see Figure 16(b)). Since Google also connects with JapanISP on the US west coast, we suspected that JapanISP was handing packets back to Google there. This was confirmed by analysis of router flow records. We believe this situation is a

```
1.  1.1.1.3                  0.339 ms
2.  1.1.1.4                  0.523 ms
3.  1.1.1.5                  0.670 ms
4.  japan2.nrt.google.com    0.888 ms
5.  exchangepoint.jp         1.538 ms
6.  router.japanisp.jp       117.391 ms
```

(a)

```
PING exchangepoint.jp
64 bytes from address2: ttl=252 time=1.814 msec
Estimated reverse path length = 4 hops

PING router.japanisp.jp
64 bytes from address3: ttl=248 time=102.234 msec
Estimated reverse path length = 8 hops
```

(b)

Figure 16: Data used in troubleshooting Case 4: (a) Extract of traceroute, and (b) Pings to routers at peering link.

legacy of the fact that Google's presence was originally limited to California, and that many Asian ISPs consider Google to be only present in the US and prefer to send traffic to all Google nodes via their PoPs in the US.

2.6.2 Resolution Of Cases

We presented the diagnostic information from *WhyHigh* for these and other similar instances of path inflation to the network operations staff at Google.

To solve the reverse path inflation back from JapanISP, Google's network admins restricted the routing announcement for the Google node in question to be available only in Japan by advertising more specific prefixes at the the PoP in Japan where Google peers with JapanISP. This change reduced the RTT from over 100 milliseconds to a few milliseconds on the paths to JapanISP we were troubleshooting. Also, *WhyHigh* reported that the entry and exit points for paths from JapanISP

	#Prefixes	#AS Paths	#ASes
Total	97852	23871	18754
Inflated	11862	1891	1510
Circuitous Fwd. Path	3865	1215	1038
Circuitous Rev. Path	4784	1579	1248
Misconfiguration	876	76	65
Limited Bandwidth	439	82	47
Lack of Peering	3776	710	456
Traffic Engineering	7410	1121	644

Table 1: *WhyHigh*'s classification of inflated paths.

were both now in Japan. Another option to address this case would have been to influence JapanISP to perform early-exit routing at their PoP in Japan, instead of having a higher preference to route traffic to Google via the US.

The network admins informed us that PhilISP2 advertises a less specific prefix to Google than to its other neighbors because of the limited capacity on the peering link between Google and PhilISP2. Once this link was upgraded, *WhyHigh* no longer detected inflated paths to PhilISP2.

2.6.3 Summarizing Use Of *WhyHigh*

Table 1 presents a breakdown of the various causes into which *WhyHigh* classified the latency inflation problems observed in our RTT dataset. The number of cases to be investigated reduced by almost an order of magnitude when considering the set of affected AS paths and ASes, rather than prefixes. While most of the cases were diagnosed as being due to lack of peering or traffic engineering, there were also a significant number of cases affected by routing misconfigurations and limited bandwidth on peering links.

We analyze one of the causes—lack of peering—in more detail. Given a node with inflated paths, we seek to understand what fraction of these instances of inflation are in ISPs reached via transit ASes versus those in ISPs with whom Google

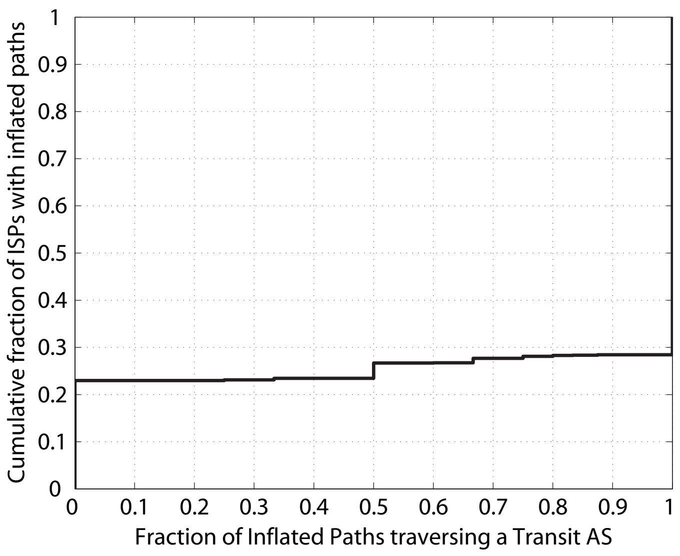

Figure 17: Fraction of inflated AS paths to an AS that goes via a transit AS. At this particular node, around 70% of ISPs experiencing path inflation have all paths going over transit, indicating that peering could be a potential solution.

peers. This is important because peering relations with ISPs are expensive to establish and should be utilized efficiently. Also, if an ISP with significant traffic has inflated latencies because of lack of peering, i.e., the path traverses multiple transit providers, it maybe worthwhile to establish new peering relations.

Figure 17 plots the fraction of inflated latency AS paths to an ISP that traverse a transit provider. A path is considered to have inflated latency if all the prefixes served by that path experience latency inflation. We see that for this particular node, around 70% of ISPs experiencing path inflation have all their inflated AS Paths via transit providers. Lack of peering relations is the likely reason for inflation on paths to ISPs in this set. We also see that around 22% of the ISPs with inflated paths are so even though all paths utilize the direct peering with Google. These cases are important to debug because significant time and effort is involved in establishing peering relations. Also, it is easier for administrators to contact peer ISPs to fix misconfigurations in their network.

Similar to the example cases outlined above, several other instances of inflation have been fixed, and the resolution of many instances we diagnosed is in progress.

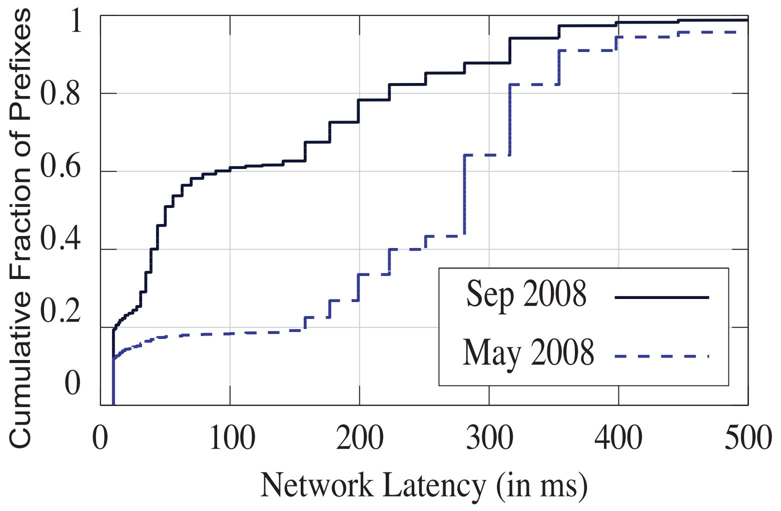

Figure 18: Improvement in latencies for prefixes served by a node in South America from May 2008 to August 2008.

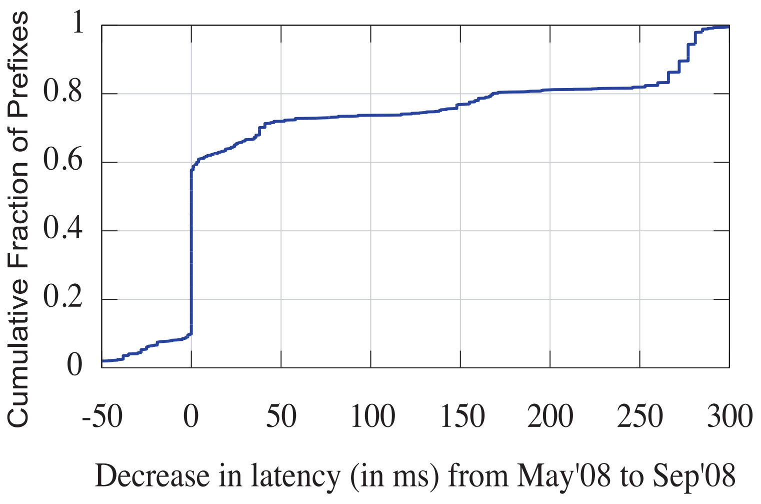

Figure 19: Change in latency of specific prefixes for the node in South America from May 2008 to August 2008

```
1.  1.1.1.6                     0.186 ms
2.  1.1.1.7                     0.428 ms
3.  node.seasia.google.com      1.727 ms
4.  router1.nrt.asianisp.net    35.600 ms
5.  router2.nrt.asianisp.net    35.644 ms
6.  router.tpe.taiwanisp.net    513.077 ms
```

Figure 20: Extract of traceroute from an inconclusive case of inflation observed from a Google node in Southeast Asia to a destination in Taiwan.

Since several thousand prefixes are affected by inflation, we have been prioritizing cases in the order in which Google receives traffic corresponding to them. To provide an example of the utility of *WhyHigh*, Figure 18 presents the improvement in path latencies for prefixes served by a Google node in South America—a node where we have focused a lot of our efforts at troubleshooting since the extent of the problems here were significantly worse compared to elsewhere. In May 2008, as many as 80% of prefixes that were being redirected to this node were experiencing latencies more than 50ms. This dropped to around 50% in September. Figure 19 shows latency changes for specific prefixes in this period. Although less than 10% of prefixes have experienced some latency increase, almost 30% of prefixes have experienced decrease in latency over 50ms and nearly 25% of prefixes have experienced a decrease in latency over 100ms. The information provided by *WhyHigh* has helped Google's network operations staff to work with ISPs in the region and significantly improve path latencies. More recent data from August 2008 shows that the fraction of prefixes being served by this South American node that suffer over 50ms of inflation is now reduced to 22%.

2.7 Limitations

After having looked at examples in Section 2.6 of cases that *WhyHigh* was successfully able to diagnose, we now consider an example where *WhyHigh* was inconclusive. We observed RTTs in excess of 500ms from a node in Southeast Asia to a bunch of prefixes hosted by an ISP in Taiwan, which does not peer with Google.

The traceroute to an address in one of the problematic prefixes (shown in Figure 20) revealed that the path traversed another ISP in Asia enroute to the destination ISP in Taiwan. Note that RTT from the 5^{th} hop in Narita, Japan to the 6^{th} hop in Taipei jumps from 35ms to 513ms. The distance from Narita to Taipei does not account for such a steep rise in RTTs. Therefore, we expected inflation to be on the reverse path back from the destination ISP in Taiwan. However, we observed no significant jump in return TTL values in pings to successive hops on the traceroute, and we found no flow records to confirm a circuitous reverse path. As a result, we lacked sufficient evidence to confirm our suspicion of inflation along the reverse path.

We present this example to illustrate that though *WhyHigh* helps in diagnosing the cause of several instances of path inflation, it is not perfect. This is a consequence of *WhyHigh*'s partial view of Internet routing. A predominant reason is the significant presence of asymmetric paths in the Internet [39,79]. Though we were able to partially tackle route asymmetry by using flow records from routers, in order to more precisely troubleshoot problems, *WhyHigh* needs the ability to gather information about the reverse path back from clients to Google's nodes. An option is to own a measurement node within the client network which, if possible, is an expensive alternative. Reverse traceroute [52] could also prove to be useful.

WhyHigh's incomplete view of the network also stems from the fact that the finest granularity of information it uses is from traceroute. Traceroutes yield path information only at the IP routing layer, i.e., layer 3 of the networking stack. However, path inflation could occur below layer 3, e.g., in MPLS tunnels, and hence may not be explainable by the geographic locations of traceroute hops, as seen in the example presented in Figure 20.

Beyond limitations in topology information, *WhyHigh* is constrained in the kinds of problems it can troubleshoot by the fact that it only has access to RTT data. TCP transfer times of medium to large objects could be inflated by other factors such as loss rate and bandwidth. For example, even if the RTT provided by the network is low, clients behind low capacity access links such as dialup and DSL can incur high download times due to the queueing or loss of packets at the network's edge.

2.8 Conclusions And Future Work

Replicating content across a geographically distributed set of servers and redirecting every client to the server with least latency is commonly expected to significantly improve client performance. However, using data from Google's CDN infrastructure that serves over 170K prefixes across the Internet, we find that routing inefficiencies and packet queueing can greatly undermine the potential improvements in RTT that a CDN can yield. To aid administrators in the task of debugging instances of high RTTs, we have built the *WhyHigh* system which first identifies affected prefixes, then prioritizes the several tens of thousands of inflated prefixes, and finally diagnoses the causes underlying these cases. We see significant improvements in client latencies offered by Google's CDN from the use of *WhyHigh* over the course of a year.

Given the large fraction of prefixes affected by path inflation issues, we are working on better visualization of our RTT data; clustering of problems with the same underlying cause is vital in our effort to debug Internet routing. Many questions still remain open on the cause for poor latencies we observe from clients to nearby nodes, e.g., "where are packets being queued to result in the congestion overhead we observe?", and we look to answer these in the future to further improve client performance.

Chapter 3

Localization

Localization is an important aspect of identifying routing anomalies. Geographically circuitous paths can be identified by identifying router locations. Identifying the latencies of clients from the same region helps us derive base cases for routing performance. So a location service for both end user IP addresses and intermediate routers is very useful in a variety of applications.

Certain automatic methods of obtaining locations may require special hardware such as GPS [42], GSM, 802.11 WiFi Access Points [59] or other special hardware [12]. These techniques are not applicable in a general setting and cannot be used to localize intermediate routers in the network. Several previous works [36, 51, 104] have addressed the problem of identifying coarse grained locations of Internet hosts using network measurements. However, these techniques still experience significant localization errors for certain targets. In this chapter we focus on identifying the causes of these errors and provide solutions for mitigating them.

For measurement based IP geolocation, there are a set of landmarks with known locations. They can issue probes to a given target host and collect round-trip delay data. In many cases the delay measurements correlate with the distances between the two communicating parties. Thus using the measured delay we can estimate the location of a target.

Mathematically this problem formulation is the same as localization problems in other settings, such as sensor networks or WiFi networks, and there are easy solutions like triangulation[1]. But IP geolocation has a number of unique challenges because of inflated network paths, measurement errors and lack of sufficient constraints.

Existing algorithms use tools such as PING and TRACEROUTE to gather latency information. PING gives end to end latencies and TRACEROUTE gives both end to end latencies and latency to each hop along the forward path from a set of landmarks to the target. Both these tools have certain limitations – PING latencies can be inflated if the network path is circuitous, TRACEROUTE only shows hops along the forward path and can give wrong hop latencies if the reverse path is assymetric. In addition TRACEROUTES can contain multiple entries for different interfaces of a particular router which have to be merged to get a more accurate picture. Previous works [51,104] have concentrated on techniques to extract consistent data from TRACEROUTE and PING measurements. These still have limitations since these techniques are based on heuristics and hence the extracted contraints can have errors. Additional location hints such as DNS records and WHOIS database are also used to validate the extracted constraints.

In this chapter we observe that not all constraints improve accuracy and having constraints which don't contribute to the solution increases the execution time and reduces the accuracy of the results. In this chapter we use concepts from Rigidity theory to identify the correct set of constraints that can reduce localization errors. We also identify nodes with potentially low localization errors. These nodes are used as secondary landmarks to improve accuracy of simple online schemes.

In the next section we examine the existing state of the art algorithms for IP localization and describe techniques to choose the correct set of constraints.

[1]Each host measures the distance to three fixed landmarks/anchors and is located at the intersection of the circles centered at these landmarks with radii equal to the corresponding distances.

3.1 Related Work

There has been several previous works on localizing hosts on the Internet and other emerging networks such as Wireless and Sensor networks. In this section we briefly examine these works and identify areas for improvement.

3.1.1 Localization In The Internet

Most previous work for localization in the Internet use tools such as TRACEROUTE and PING to measure latencies from different landmark nodes to a target. TRACEROUTE provides richer information by also identifying routers along the forward path to the target. Another tool used to find the location of internet routers is undns [96]. This tool uses ISP naming schemes and decoding DNS names of routers to determine the router's metro location. However, this tool has two limitations - (i) it needs manually input naming rules for all ISPs which is difficult because they do not typically reveal their naming conventions, or have geo-location encoded in their dns names, and (ii) if administrators make errors in naming their routers, the tool gives wrong estimates of router locations.

Some of the earliest work in this area used simple schemes to identify the approximate location of the target host. IP2Geo [78] proposes three different techniques for localization. The GeoPing technique locates a target by mapping it to the most representative landmark. Each target probes all landmarks to build a delay vector. This delay vector is compared to similarly constructed delay vectors created by different landmarks. The target is assigned the Geo location of the landmark closest to the target in the N-dimensional delay space. This scheme works fairly well if landmarks are located close to targets. The error in localization of a target in this case is directly proportional to the distance to the closest landmark. In practice it is not possible to distribute landmarks close to all targets and better localization accuracy is desirable.

In the GeoTrack technique traceroutes are performed from different landmark nodes. The locations of intermediate routers are guessed based on their DNS names and the target is assigned to the location of the nearest localizable router. The error in this case is proportional to the distance of the closest localizable router and the accuracy of DNS based schemes. Not all ISPs include metro/pop locations of the router into their DNS names and in some cases these records can be erroneous.

The final technique proposed is called GeoCluster. In this scheme partial Geo information from sources such as WHOIS, DNS and other websources is used in conjunction with BGP data to come up with clusters of IP addresses which are likely to be colocated. However these information sources are not very accurate and some of the web sources used are proprietry and not accessible in general.

Constraint based Geolocation (CBG) [36] proposes using a multilateration scheme to identify the location of the target. Each landmark draws a circle with the radius as the estimated geodesic distance to the target. The target is assigned to the centroid of the intersection region of the circles from different landmarks. The main limitation of this work arises from the challenges in mapping latency values to geodesic distances. Several works have used results from [80], which state that digital information travels on fiber optic cables at 2/3 the speed of light. This scale factor is however a very loose upper bound. In practice the data on the internet travels at much slower speeds because of congestion along the path, latency on intermediate routers and because of geographically circuitous paths. To overcome this limitation CBG uses the inter-landmark latency to distance mapping to establish a tighter upper bound. They also show that as long as the distance is an over-estimate their technique yields good results. However in practice the computed distance to latency mapping between landmarks may not extend to specific targets, especially if targets are far from the landmarks. Hence in many cases the circles fail to intersect [51].

Both GeoPing and CBG use end to end latency information to identify the location of the target. However TRACEROUTE provides richer path information which can help constrain the target further. We now look at some schemes that greatly improve results by making use of the path information.

Topology based Geolocation (TBG) [51] uses both end to end latency information from landmarks to targets, and latencies to intermediate routers along the path to the target. These constraints are input to a Semidefinite program (SDP) based global optimization algorithm that minimizes average errors of both the routers and the targets. This technique yields high accuracy even for targets far from landmarks. However it is computationally very expensive and hence cannot be performed on-demand. Also since SDP is a global optimization, this technique can yield poor results in the presence of measurement errors.

Octant [104] is an extention of CBG but instead of identifying the target location as the centroid of the intersection region it defines the entire intersection region using bezier curves. The region can then be used as a landmark in localizing other nodes. This could potentially lead to a large number of potential regions for a target. To prune the solution they make use of both positive and negative constraints. Negative constraints are regions corresponding to a landmark where the target cannot reside. They Octant scheme performs in near real time while ensuring reasonable accuracy. Region-based methods are also more intuitive than optimization-based approaches as in [51]. This scheme can be further improved by incorporating some principles from graph theory which can help identify poorly constrained graphs.

Errikson et al [29] have cast the IP geolocation problem as a machine learning based classification problem. They selected a subset of target nodes for training a Native Bayesian classifier, with the training set nodes having both known measurements to the landmarks and known geolocation. Additional information such as location hints and population density are used to refine their framework. A target is assigned a location based on a node with similar characteristics in the training set. This technique works well in cases where there are sufficient location hints.

3.1.2 Localization In Wireless And Sensor Networks

There is a large body of work on localization of network nodes in Indoor wireless networks, Sensor Networks and Mobile Ad-hoc networks. Algorithms similar to the ones used in the Internet have been used for localizing nodes in Wireles and Sensor network. In this section we only discuss some work which has inspired similar work in Internet localization.

The RADAR system [12] localizes users in an indoor wireless setting. They first create a signal strength map of the indoor area. This map contains the x, y location of a test user and the signal strength seen by 3 different base stations for that user for that location and orientation. This is collected in an offline process. For localizing a user, the signal strength at observed at different base stations is compared against the offline data and the user is assigned the nearest point in signal space. This technique is very similar to the one used by GeoPing.

Sextant [37] is a precursor to Octant and is used to localize nodes in Sensor networks. They use Bezier curves to define potential regions where a target is located and make use of positive and negative constraints to reduce the size of this region. Positive constraints identify where a node could be located based on message transmissions from neighbors. If a node cannot receive transmissions from one of its neighbors, the neighbor cannot exist within a smaller region around the node and this information can be used as a negative constraint.

In [61,103] authors propose an anchor free localization algorithm for sensor networks. They select a subset of nodes as landmarks and extract the combinatorial delaunay complex of the graph. Delaunay complex of a graph maybe globally rigid even if the underlying graph is not thereby avoiding flip ambiguities in localization which arise from insufficient constraints. The authors prove this for the case where the landmark set is sufficiently dense. A target is localized using landmarks in ajacent. They then use an incremental algorithm to glue these delaunay triangles together and thereby localizing the landmarks. All other nodes in the graph are localized using triangulation with respect to their closest landmarks.

3.2 Background

In IP geolocation, we make use of latency measurements between two hosts on the Internet to help locate a target host. The measured latency can be translated to distance using the following heuristics. Data travels through fiber optic cables at almost 2/3 the speed of light in vacuum. Due to packetization and other non propagation delays, the real speed observed on the Internet is around $4/9$ the speed of light [51].

There are two types of latency information we can obtain – the end-to-end roundtrip delay (RTT), or more detailed delay information for each segment of the routing path. End to end delays can be *very noisy*, due to the circuitousness and irregularity of Internet paths [96]. Network traffic dynamics and queueing effects at the routers also add to the variations of the delay measurements.

Recognizing that a routing path between two hosts consists of multiple links through other routers, one can make use of the network topology to improve the geolocation accuracy. Using the TRACEROUTE command, one can get an estimated delay on each link of the path. This gives a graph G on the targets, the landmarks, and the intermediate routers on the paths from targets to the landmarks. All the edges of this graph are given some estimated length. We solve for the positions of the routers *together* with the location of the targets. The problem thus becomes *network localization*. This is the same as the multi-lateration problem [90] in sensor network literature and is named *topology-based geolocation* in [51].

We classify existing IP geolocation solutions by two categories: those that only use round-trip delay to landmarks with known locations are called *triangulation-based* methods; those that also use the intermediate routers and detailed estimated delay on each link of the routing path are called the *topology-based geolocation* methods. Triangulation-based methods one solves the target location by using a small number of constraints and the solution is typically obtained by solving a constant size problem. Topology based methods may use intense optimization routines such as semi-definite programming. Generally speaking, triangulation-based methods are fast but less accurate; topology-based methods are more accurate but slow.

We briefly review some of the representative methods in each category as our solution is based on both.

Shortest Ping. A target is localized to its closest landmark node. This is a very simple scheme that maps each target to its closest landmark and is used as a baseline for other simple schemes. Clearly the localization error is directly dependent on the distance to the closest landmark. A similar scheme, GeoPing [78], assumes that hosts with similar network delays with respect to other landmarks are located near each other. Thus a target node is given the geographical location of the landmark x if its delay profile to other landmarks is most similar to that of x. The resolution of GeoPing also heavily depends on the density of landmarks.

Constraint Based Geo-location (CBG). A target is localized to the centroid of the intersection region of circles centered at different landmarks [36]. Every landmark is associated with a scale factor based on the delay to distance ratio from that landmark to a set of nodes with known location (including other landmarks). The scale factor is the maximum value of distance / delay ratio. Due to measurement errors sometimes intersection regions may not exist. In this case the target is assigned to some point on the boundary of its nearest landmark. Typically the error in this technique is of the order of the distance to the nearest landmark. As an extension of the scheme, Octant [104] incorporates both upper bound and lower bound distance constraints and describes the bounded region using Bezier curves. Region-based methods are more intuitive than optimization-based approaches as in [51], described below.

Topology based Geo-location (TBG). TBG [51] uses both end to end latency information from landmarks to targets, and latencies to intermediate routers along the path to the target. These constraints are input to a semi-definite program (SDP) based global optimization algorithm that minimizes average errors of both the routers and the targets. This technique is computationally very expensive – it takes approximately *one week* to localize the 128 targets in our dataset with additional 500 intermediate routers. This scheme yields the best accuracy among all existing solutions, and in particular gives better performance for targets far from any landmarks.

3.2.1 Challenges Of IP Geolocation

With the existing solutions, we examine the algorithm and implementation challenges for IP geolocation:

3.2.1.1 Challenge I: Measurement Errors

Given a target host with RTT latency to a set of landmarks with known locations, robust localization algorithms that can tolerate measurement errors or outliers are largely missing.

Optimization based solutions are very sensitive to measurement errors. For example, a routine procedure in many network localization algorithms is 'mass-spring' relaxation, where the nodes are treated as masses and edges are treated as springs with target lengths as the given measurements. The locations of the nodes are adjusted so as to minimize the 'spring energy'. While small, random errors in the input can be nicely smoothed out, large errors are spread all over the network leading to worse solutions. Large input error may also cause conflicting constraints (e.g., triangular inequality could be violated). Thus global optimizations such as semi-definite programming [15] may fail to find a feasible solution.

For region-based schemes [36,104], each constraint gives a feasible geometric region for the target location. Incorporating different constraints means taking the common intersection of these feasible regions and the final location is often estimated as the center of mass of the feasible region. Again, these methods can break down when there are large measurement errors. Even with one poor measurement, the feasible regions may have an empty common intersection. The sensible thing to do is to first filter out measurements with large errors. Unfortunately, how to filter out bad data in cases of conflicts is not known.

3.2.1.2 Challenge II: Incorrect/Insufficient Constraints

In topology-based solutions, an Internet graph with the target hosts and intermediate routers are induced. An important issue observed in our experiments with real data is that the solution for the positions of the routers and targets in G may not be unique. That is, the distance constraints maybe insufficient to uniquely determine a solution. When there are multiple solutions, an algorithm may arrive at a solution which is still far from the ground truth. Whether or not G has a unique realization is a problem in *rigidity theory* [35]. In rigidity theory, one formulates a bar-and-joint framework, where each distance constraint represents a rigid bar connecting the two vertices and asks whether the distance constraints are sufficient to remove the flexibility of the framework. The graph is called globally rigid if there is only a unique realization in the plane subject to rigid motions. Thus we need to have at least enough constraints to ensure a globally rigid graph (See Figure 21).

Having too many constraints is not necessarily a beneficial thing either. The induced Internet graph could also be over-constrained in some regions and these constraints increase the chance that some conflict with each other. In this case it would be helpful to reduce the problem size by pruning out certain constraints. But how to prune and which edge to exclude would also require knowledge in graph rigidity. None of the existing work in IP geolocation has examined graph rigidity yet. In particular, how much error in the solution is caused by improper or insufficient constraints and how this problem can be mitigated, is still to be investigated.

3.2.1.3 Challenge III: Efficient Algorithm Design

Triangulation-based algorithms using RTT delay information are fast but inaccurate. Topology-based IP geolocation algorithms are accurate but slow. In addition, semi-definite programming is a 'blackbox' solution. It is hard to investigate the source of error. Understanding the sources of error is important to improve the location accuracy using simpler, faster algorithms. This is the motivation in our study and our objective is to develop algorithms as fast as triangulation-based solutions, with accuracy comparable to that of topology-based IP geolocation algorithms.

In the rest of the chapter we first review some basic notions of rigidity theory. Then we present error analysis of existing IP geolocation solutions using real data sets. With the observations from real data sets we will present our solution by using *2-tier landmark scheme* and *rigidity-based constraint pruning*.

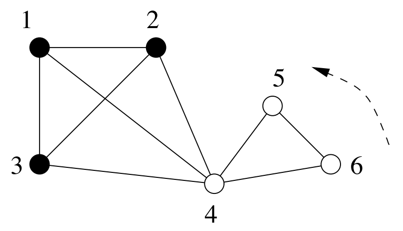

Figure 21: A graph can be broken down into rigid subgraphs. The dark dots are landmarks. In this case nodes 1,2,3 and 4 belong to a pinned rigid subgraph with a unique realization. Although sub-graph 4,5,6 is rigid, it is not globally rigid – nodes 5 and 6 can have infinitely many realizations while satisfying the distance constraints.

3.3 Rigidity Theory

Given a set of bars connected by convolute joints, rigidity theory studies whether the framework can be deformed or not [35]. A graph is considered rigid in 2D if it cannot be transformed into a different configuration with the same edge lengths through non-trivial (e.g., excluding global translations and rotations) smooth motion within the plane. A graph is globally rigid if it has a unique realization in the plane, subject to global translations and rotations.

The rigidity of a graph can be understood by examining the number of degrees of freedom (DoFs) it has. A globally rigid graph only has planar realizations that are rigid transformations of one another. Thus there are only three degrees of freedom, corresponding to two DoFs of translation and one DoF of rotation. When any one node and the direction of an edge are fixed, the entire realization is fixed and unique. For a rigid graph and one of its realizations in the plane, the only smooth motion is rigid motion as well. But there are probably multiple non-congruent realizations of the graph, although one cannot use smooth motion in the plane to convert one to the other.

Graph rigidity in 2D is well understood. The family of rigid graphs admits a combinatorial condition described by the Laman Theorem [58], which states that a graph with n vertices is rigid in 2 dimensions if it has $2n-3$ edges and iff $e' <= 2n'-3$ for every subgraph of G having n' vertices and e' graph edges.

Therefore, rigidity is a combinatorial property rather than a geometric property. It is purely determined by the number and position of edges rather than the edge lengths. Intuitively this is because each edge places one constraints on the positions of the endpoints and the the number of edges in a graph are the number of constraints or restrictions on the degrees of freedom. There is also efficient algorithms, called the pebble game [45], to test whether a graph is rigid or not. In case that the graph is not rigid, the pebble game can identify the rigid components.

In our setting we have a graph with some nodes as landmarks, i.e., with known locations. This is referred to as pinned vertices in rigidity theory and can be incorporated in a similar Laman condition. Basically a pinned vertex fixes two degrees of freedom. The pebble game for finding rigid subgraphs can be generalized to include pinned vertices [63,64]. This is illustrated in figure 21. The subgraph 1-2-3-4 is globally rigid, while the entire graph is not rigid (the subgraph 4-5-6 is flexible with respect to the rest of the graph).

The localization problem can be thought of as finding the realization, i.e., pinning down the graph in the plane such that it has 0 degrees of the freedom. In particular, given a graph $G = (V,E)$, together with a set of non-negative edge weights ℓ_{ij} (for all edges $(i, j) \in E$), the goal is to compute a realization of G in the Euclidean space $\mathbb{R}^d$ for a given dimension d, i.e. to place the vertices of G in $\mathbb{R}^d$ such that the Euclidean distance between every pair of adjacent vertices (i, j) in E equals the prescribed weight ℓ_{ij}. We focus on $d = 2$. A flexible graph can have many configurations and hence cannot be pinned down or localized. A rigid graph may still yield a valid realization that is different from the ground truth, if there are many non-congruent valid realizations. Hence, for a graph to be completely localizable it has to be globally rigid at least [28], [8]. Combinatorial conditions and efficient algorithms for testing whether a graph is globally rigid or not are also available. In particular, A graph is uniquely realizable under edge lengths constraints if and only if it's tri-connected (the graph is connected even if two vertices are removed) and is redundantly rigid (a graph remains rigid upon removal of any single edge) [13,41]. For a globally rigid graph the realization is unique but has the freedom of a rigid motion. For IP geolocation, we require at least a globally rigid graph and a set of at least three landmarks to uniquely pin down the graph.

Although the conditions for rigidity or global rigidity have been fairly understood, efficient algorithms for computing the realization do not exist. To understand this, we can formulate the problem of network localization as 'matrix completion' problem, that is, completing a distance matrix with some entries already specified by the edge weights such that the distance matrix corresponds to the pairwise Euclidean distances of all the vertices places in $\mathbb{R}^d$. It is known that a set of pairwise distances arise from points in $\mathbb{R}^d$ (but not $\mathbb{R}^{d-1}$) if and only if a certain matrix is positive semi-definite and has rank d [92, 106]. In this case, we also say that the edge weights admit a positive semidefinite completion with rank d.

If we want to have a realization of a graph in some Euclidean space and do not care about the dimension, the problem is easy. It is known that with semidefinite programming graph realization problem is solvable in polynomial time if the dimension of the realization is not restricted. However, if we restrict the dimension of the realization to be exactly d for some fixed d, this rank constraint makes the problem to be non-convex. Finding a realization in 2D given edge length is NP-hard, even if the graph is globally rigid [9, 11, 91]. This negative result, nevertheless, does not rule out the possibility of polynomial algorithms for realization of special graphs, in particular, the *uniquely d-localizable* graphs.

A graph is *uniquely d-localizable* if for some generic edge lengths ℓ_G, there is a unique realization in $\mathbb{R}^d$, then there is no non-trivial realization in $\mathbb{R}^k$ for $k > d$. That is, there is no other non-congruent realizations even if we allow the realization to be in higher dimensions. Thus the rank constraints can be removed and the network localization problem is solvable by semidefinite programming. For uniquely d-localizable graphs, So and Ye proved that a SDP algorithm solves the realization precisely [15, 94]. The intuition is that for uniquely d-localizable graphs, one does not need to worry about the rank of the positive semidefinite completion as any matrix completion automatically has rank d by definition. This semi-definite programming solution is used in the topology-based methods [51].

However, for the partial Internet graphs constructed from latency measurements, global rigidity can be too restrictive a requirement. We would like to understand the structural characteristics of the sub-graphs of the constructed Internet graph and identify rigid and flexible regions within it. In order to do so, we use a class of algorithms called the Pebble Games [41, 59, 60]. These are described in the next section.

3.3.1 Component Pebble Game

Pebble game algorithms have been developed to identify underconstrained, overconstrained and rigid subgraphs of a graph. In this section we briefly look at one algorithm which is designed to also return the list of rigid subgraphs in a graph.

For a graph $G(V,E)$, a pebble graph $PG(V,E')$ is constructed as follows. Every node in the pebble graph is given k free pebbles corresponding to degrees of freedom. An edge e on vertices u,v from G can be added to PG if there are atleast $l+1$ free pebbles on u and v. The edge e is then inserted into PG and assigned one free pebble. If the pebble was assigned from node u the direction of the edge is from u to v. The assigned pebble is not free anymore. Here k represents local and l represents global degrees of freedom. For the 2 dimensional case $k = 2$ and $l = 3$. In the Component pebble game, when an edge is added the component associated with this edge can be identified by computing the *Reach* of the two vertices. *Reach* of a vertex u is the set of vertices in PG that can be reached via directed paths from u in PG.

One algorithm to identify rigid components is described below. More details can be found in [62]. In subsequent sections we show how these components can be used to improve the accuracy of existing localization algorithms.

3.4 Dataset

We use the University dataset made available by the previous work [51]. This dataset consists of TRACEROUTE measurements from different PlanetLab nodes.

Algorithm 1 COMPONENTPEBBLEGAME ($PG = (V, E')$)

Input: Directed pebble graph $PG(V, E')$ where Edge $e = uv$ was just added and atleast 3 pebbles are present on u and v.
Result: Component C associated with $e = uv$

```
1: $Reach(u,v) = Reach(u) \cup Reach(v)$
2: if Any $w \in Reach(u,v)$ has atleast one free pebble then
3:     return $\phi$
4: end if
5: $V' = Reach(u,v)$
6: $Q =$ Vertex in $V - V'$ with an edge into $V'$
7: while $Q$ is not empty do
8:     $w = Dequeue(Q)$
9:     if All vertices in $Reach(w)$ other than $u$,$v$ have no free pebbles then
10:        $V' = V \cup Reach(w)$
11:        $Q+ =$ Vertex in $V - V'$ with an edge into $V'$
12:    end if
13: end while
14: return $V'$
```

68 nodes at different PlanetLab sites in North America serve as landmarks and 125 hosts at universities across the US are used for targets. The targets span 37 out of 48 states on the mainland. In addition to these targets, location hints such as DNS records were used to identify the location of routers along the paths. A set of 500 additional routers with validated location information were used as a set of potential passive landmarks. Location hints were already present in the dataset, we validated them further using IPlane [70] data.

We found this dataset particularly suited for our study. Firstly, it is a real dataset and has all the limitations of real world measurements including measurement errors. Secondly, the dataset has been sterilized to remove some of the common sources of measurement errors such as the presence of assymetric network paths in TRACEROUTES and MPLS segments that distort measurements. In addition interfaces belonging to the same router have been clustered in order to avoid spurious links.

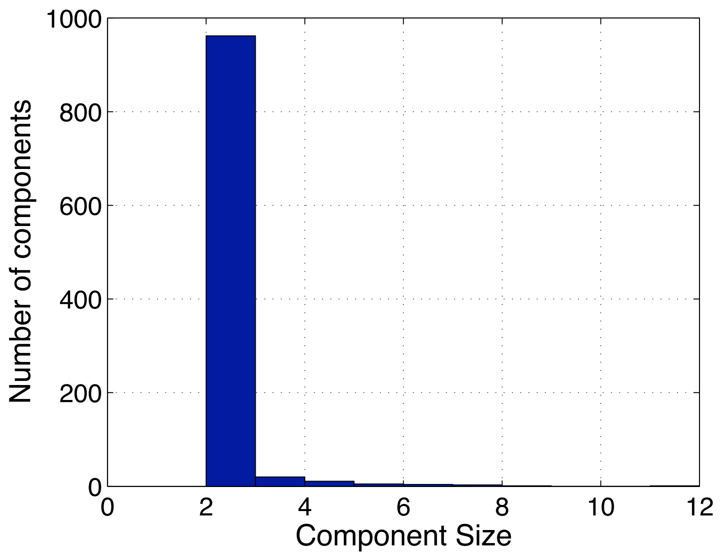

Figure 22: Pebble game breaks the input graph into components. The graph consists of one component of 868 nodes and many components of less than 12 nodes. This graph only shows the histogram of components of less than 12 nodes. Note that a node may belong to upto two components.

Some preliminary analysis on the dataset revealed that the network graph built from the dataset is fairly dense and has $n = 1912$ nodes and $m = 3733$ links. The nodes include landmarks, targets and intermediate routers.

We first test whether the graph is globally rigid, i.e., admitting only one realization in 2D. It turns out that the graph is not globally rigid. However, the graph could be decomposed into rigid subgraphs [45]. The graph consists of one large rigid sub-graph of 860 nodes. Almost 830 nodes belong to subgraphs containing 3 or lesser nodes. A histogram of sizes of components returned by the Pebble game is shown in figure 22. These nodes are likely to have high localization errors because the sub-graph containing the node is flexible with respect to the rest of the graph(21), and hence has many possible solutions.

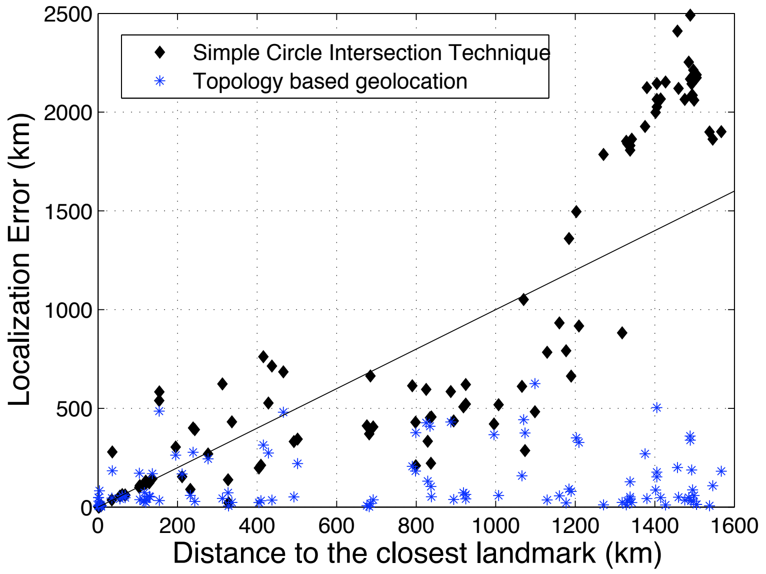

Figure 23: Localization error as a function of the distance to the nearest landmark.

3.5 Error Analysis

In this section we examine the limitations of existing geo-location schemes. For that we first examine what are the typical sources of geo-location errors using the dataset described in the previous section.

3.5.1 Error In End To End Delay Constraints

Previous works have established that simple schemes can perform quite well but are limited by the distance to the nearest landmark. To illustrate this we plot the localization error of two schemes CBG and TBG compared to the distance to the nearest landmark in Figure 23. This figure illustrates that heavy duty algorithms that use global optimization with a large number of constraints can perform very well even when the landmarks are far away.

Techniques such as CBG based on end to end latency measurement typically have errors proportional to the distance to the nearest landmark. However we see that for some targets the errors are disproportionately high (top right part of the

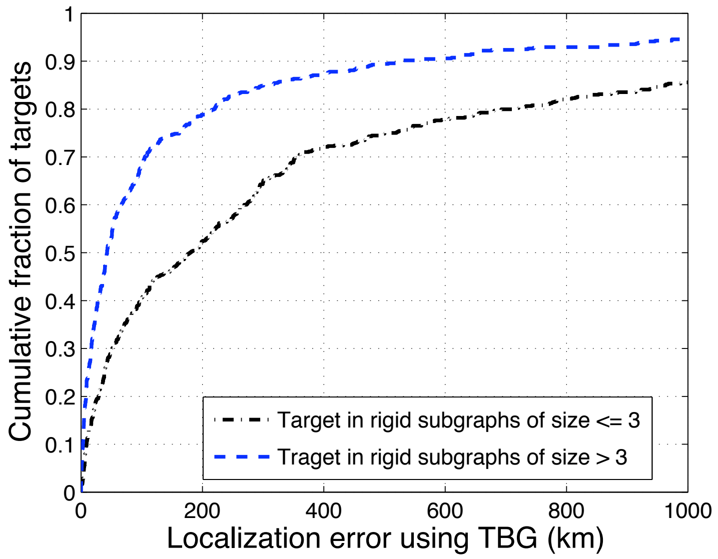

Figure 24: Localization error using TBG algorithm for two sets of nodes – nodes in rigid sub-graphs of $<=$ 3 nodes and nodes in larger rigid sub-graphs. Nodes include targets as well as validated intermediate routers

graph). This is caused by inflation in end to end latencies because of circuitous paths. This confirms that the triangulation-based solutions have better accuracy if we use only delay data to landmarks with low latencies. Indeed, the routing path to far-away landmarks is less likely to be direct and the distance derived from latency is often an over-estimate. Using such inaccurate distance measurements in triangulation is not wise as it may contradict other measurements and yield biased solutions.

3.5.2 Error In Global Optimizations

Global optimization algorithms such as SDP are a very powerful tool in finding the correct location of nodes that satisfy a set of constraints. It is important to note that not all constraints help improve the solution. In this section we examine how choosing the right constraints affects the solution.

The pebble game algorithm described in section 3.3.1 identifies if the graph is over-constrained, under-constrained or well constrained. It is also used break a

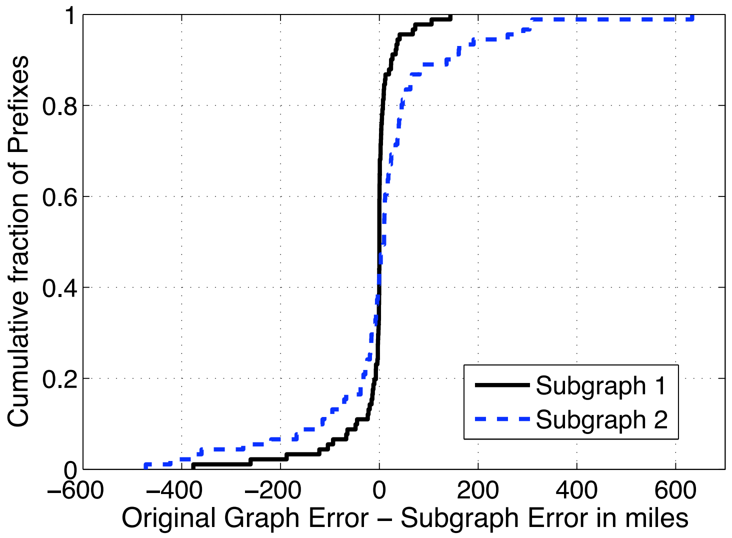

Figure 25: Two sub-graphs (Subgraph 1 and 2) of the original constraint graph G are created by removing links from over-constrained regions to make them well constrained. This graph plots the difference between running the TBG algorithm on the original graph for a particular target vs on each sub-graph. The error distribution indicates that removing constraints from over-constrained regions does not affect TBG results.

graph into rigid components. We saw that a large number of nodes in our dataset belonged to rigid components which have less than or equal to 3 nodes. Figure 24 shows the localization errors using the TBG scheme for targets in two sets – targets in rigid subgraphs of 3 or fewer nodes and targets in other subgraphs. For this evaluation, we use 125 targets as well as the 500 validated intermediate nodes.

This figure shows that even with computationally expensive optimization algorithms targets in smaller rigid subgraphs experience much higher errors than other nodes. These regions of the graph would benefit from having more constraints added to them.

Pebble game algorithms can also be used to filter out links that cause a sub-graph to be over-constrained. An over-constrained sub-graph contains more links than needed to correctly localize it. For example a graph with 4 nodes and 6 links is over constrained and any one of those 6 links can be removed to make it well constrained. Hence there are different well constrained sub-graphs associated with an over constrained graph. Changing the order of links input to the pebble game

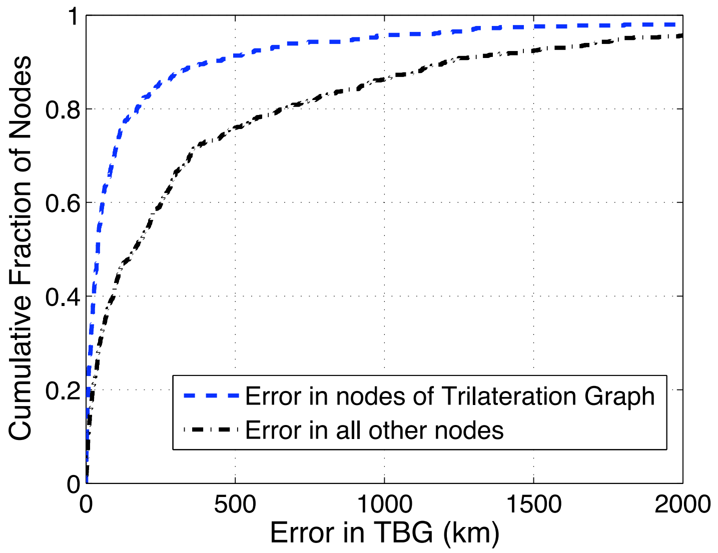

Figure 26: Localization error using Semi-definite programming for nodes in the trilateration graph vs other nodes in the dataset.

yields different sub-graphs which have no over-constrained regions. To verify if over-constrained regions cause increase in localization errors we plot the difference in localization errors for all 125 targets between running TBG on the original graph vs running TBG on a subgraph computed by the Pebble game. The graph 25 shows that the errors are almost symmetrical and these regions do not affect the performance of the algorithm.

We can conclude that rigidity alone does not help us distinguish between nodes that can be localized accurately and those that cannot. We need a stronger condition and for that we look at trilateration graphs described in the next section.

3.5.2.1 Trilateration Graphs

TBG uses SDP to solve a joint optimization problem to get the positions of a target and intermediate nodes. But the caveat is that we can only solve it correctly when the SDP formulation has a unique solution. It is known that the SDP formulation gives a unique solution only for a *uniquely* 2*-localizable* graph [94], which is

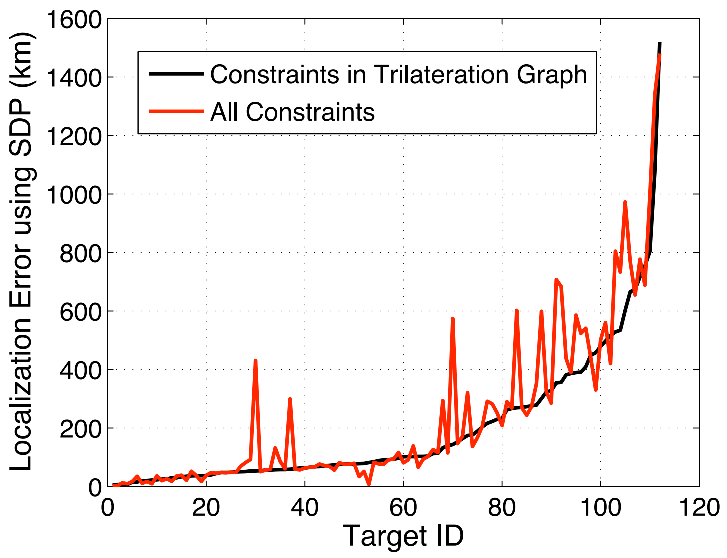

Figure 27: Localization error in TBG using only constraints from the trilateration graph vs original TBG which uses all constraints.

globally rigid in 2D and has no non-trivial realizations in any higher dimensions[2]. It is still an open problem how to test whether a graph is uniquely 2-localizable and how to characterize the family of uniquely 2-localizable graphs. The only family of uniquely 2-localizable graphs known to us is the *trilateration graph*. We call a graph to be a trilateration graph if it is either a complete graph on no more than 4 vertices or is built inductively by adding one vertex and connecting it to three existing vertices.

Given an arbitrary graph, we can use the following greedy algorithm to decompose the graph into components of trilateration graphs. To do so, we start with the original set of landmarks. These can be said to form a trilateration graph since we can create a completely connected graph for landmarks by measuring all inter landmark latencies. Then we add vertices one by one. Each time we test whether we can find a vertex with three edges to existing vertices. If so this vertex is added.

[2]Even if we take the framework in 3D or higher, the only possible realization is a rigid transformation of the unique realization in 2D.

We stop when no more vertices can be included and proceed to the SDP localization for a trilateration instance. After that, we can remove these nodes and proceed with the rest of the nodes using the same greedy routine. Eventually every node is in some trilateration instance, unless it has only one or two edges to existing sets of landmarks. For those cases we know that the node location is not unique. If a node is only connected to one landmark, it has one degree of freedom and can be placed at any position of a circle centered at the landmark with radius as the distance measured by latency. If a node has only edges to two landmarks, its position is not unique and differs from one another by a reflection. In the two cases we either identify the target with the closest landmark (causing some substantial error), or simply give up as there is not enough information.

All the nodes of a trilateration graph are uniquely 2-localizable and hence have a good chance of getting more accurate results using SDP. To verify this point we plot the error using the original TBG scheme for target nodes in the trilateration graph and for the rest of the target nodes in Figure 26. This evaluation uses the larger target set consisting of the original target set and intermediate routers with valid geo. We see that nodes in the trilateration graph have significantly less error than other nodes and hence are good candidates for new landmarks.

The SDP formulation in TBG uses both end to end latencies and link constraints extracted from the traceroutes. We modify TBG to only use those intermediate nodes which belong to the trilateration graph. For our reference we call this TBG-trilateration. The target is included even if it is not a part of the trilateration graph. So for the target, end-to-end delay constraints from the landmarks and from the intermediate nodes in the trilateration graph are included. Figure 27 shows the error in TBG vs TBG-trilateration. We see that for many targets TBG-trilateration significantly improves localization error. The number of constraints in TBG-trilateration is on average 30% less than in TBG. Running TBG-trilateration on 128 targets took around 5 days to run compared to 7 days for original TBG. This shows that choosing the right set of constraints is very important and can significantly affect localization results both in terms of accuracy and speed.

We can conclude that first, localization error is lower for targets which are a part of the trilateration graph. This can be used to identify targets with potentially low localization error and can be selected as passive landmarks. Second, using fewer constraints from trilateration graph constructed out of the input graph improves overall localization accuracy of the TBG scheme.

Thus given a set of landmarks with known locations, we can find additional landmarks from within trilateration graphs and such that all targets have small delays. This ensures that these secondary landmarks can be localized with high accuracy and simple schemes can be used to localize other targets. This leads to our 2-tier landmark structure design, to be explained in detail in the next section.

3.6 2-Tier Landmark Structure

Error analysis in the previous sections revealed that measurement errors for simple schemes largely come from indirect and inflated routing paths. To mitigate that we try to avoid using delay measurements to 'far away' landmarks.

With the given landmarks, we first look for additional landmarks that are nicely dispersed such that any potential target host has small delay to at least one landmark. The newly selected landmarks will be localized in an offline manner, using algorithms such as TBG-trilateration with the best accuracy (not necessarily with real time performance).

For real-time IP geolocation of a target host, we only use the landmarks with small delays to the target and employ triangulation based methods, which are fast. This way we strike a nice balance between efficiency and accuracy. In the following we discuss landmark selection, landmark localization and target localization respectively.

3.6.1 Landmark Selection

The goal is to select landmarks that are nicely dispersed on the Internet to 'cover' all target hosts. There are different ways to select additional landmarks. One obvious scheme is to select these landmarks by *geographical locations* such

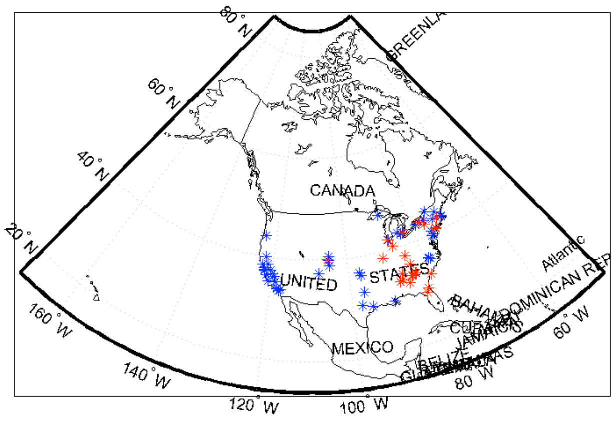

Figure 28: Geographic Distribution of new landmarks added based on latency data. Red points represent the original landmarks while the blue points represent the new landmarks

that all hosts are geographically close to at least one landmark. However, due to irregularities of routing paths, the smallest delay to landmarks may not decrease much with the addition of new landmarks based on geographical distribution. In a recent observation in content delivery networks (CDN) [55], around 30% of prefixes in a CDN have high latencies (> 50ms) to geographically closest nodes, and 20% of all prefixes have inflated path latencies. These results revealed the surprising level of inflation in routing delay, introduced by network topology and protocol-level complexities.

The second method is to select a well dispersed set of passive landmarks in the *delay space* to ensure all hosts are close to at least one landmark measured by delay. Understanding the Internet delay space has been a research subject for some time. Multiple methods have demonstrated that one can fairly well embed the Internet in a low dimensional Euclidean space. A prominent one of these is the Global Network Positioning project (GNP) [77] which demonstrated that one can predict the Internet delay for the majority of host pairs by using an embedding in a 7-dimensional Euclidean space. With this property we can exploit the geometric packing argument to select uniformly spread landmarks. For example, if we would like to ensure that all hosts are within delay d from at least one landmark, we execute the following greedy algorithm, in which we select a new landmark that has longest delay from existing landmarks until the condition is satisfied. This method ensures that the newly added landmarks are not cluttered with existing landmarks and thus are likely to cover a 'remote' part of the delay space.

We construct the Internet delay graph from the traceroutes in the dataset and use that to identify a set of well dispersed nodes in delay space. This does not give us the exact latency between a pair of nodes, but since the graph is fairly dense we use the shortest path latency between nodes in the graph.

Choosing landmarks solely based on delay may have an adverse impact, as a node with high delay from existing landmarks is also likely to be on the periphery of the Internet and is poorly connected to existing landmarks. Poor connectivity could result in high errors for global optimization algorithms. We want to choose these landmarks such that not only are they well dispersed in delay space, they also have small localization errors using traditional optimization techniques. We have seen in the previous section that using constraints in a trilateration graph reduce localization errors and targets that are a part of the trilateration graph experience less localization error. Hence we use these nodes as candidate set for secondary landmarks.

We use one more criterion to choose new landmarks. We give higher priority to passive landmarks which are on the path to a larger number of targets. Thus more targets can be affected by the addition of fewer landmarks.

The landmark selection algorithm for a single passive landmark then becomes:

- We only select passive landmarks from nodes in the trilateration graph.
- For each candidate node i we compute the sum of latencies from existing landmarks to this node and denote this LS_i. Latency between any two nodes in the trilateration graph is computed based on the latencies along links in the shortest path of the graph. We record $LS_{max} = arg_{max} LS_i$.

- Of nodes that have $LS_i > 0.9 \times LS_{max}$ we choose the node that appears in TRACEROUTE paths to a maximum number of targets and add this node to the set of landmarks.

This process of landmark selection is repeated for the number of required passive landmarks. Figure 28 shows the geographic distribution of 60 landmarks selected based on this scheme. We computed the minimum latencies from all targets to these passive landmarks, and the median of these min latencies is 7.1 ms while the median of median latencies from all targets to these landmarks is 35 ms. This shows that this scheme is able to select landmarks which are not only geographically dispersed but also distributed in terms of latencies to targets.

For comparison we also computed a set of passive landmarks based on just *geographical distribution*. The median of minimum latencies from all targets to these landmarks was 13 ms. This does not seem like much of an increase till we note that 6 ms increase corresponds to distance of 750 km if we use $4/9$ speed of light as the conversion ratio.

To localize these new landmarks we use the TBG-trilateration algorithm as shown earlier. To localize a particular node we use all the links on the path from the original landmarks to the new node in the trilateration graph.

3.6.2 Target Geolocation

Although topology-based geolocation requires active landmarks, i.e., hosts that can issue TRACEROUTE probes, the newly added landmarks are potentially only passive landmarks, i.e., they cannot issue probes to the target host. These passive landmarks have been used by many of the previous works to improve localization. For online localization targets can easily send probe packets to these landmarks which can then be used to triangulate them. Passive landmarks that are on the traceroute path to the target can easily be incorporated in simple delay based schemes to localize a target.

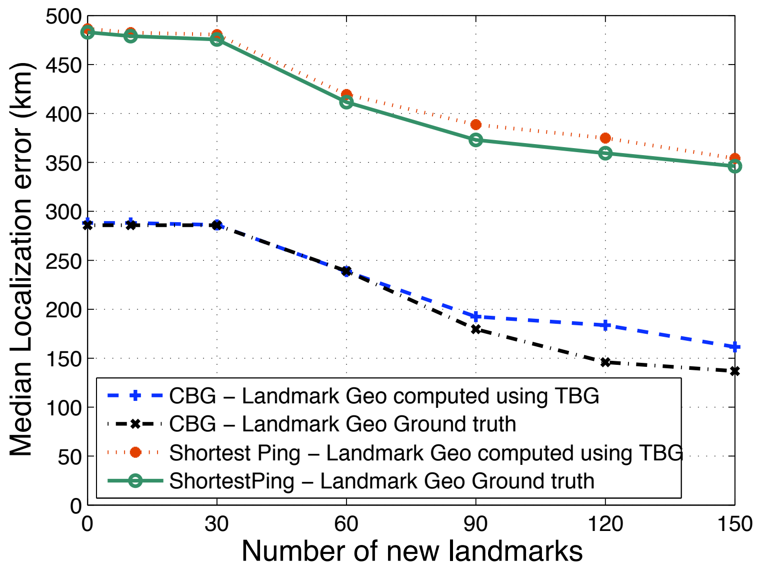

Figure 29: Median Localization error for all targets using CBG and ShortestPing schemes as more passive landmarks are added. The graph plots the median error when the location of new landmarks is computed using TBG-trilateration vs using ground truth for the location. We see that using TBG-trilateration affects median error by less than 20kms when 90 additional landmarks are used.

3.6.3 Experimental Results

We use the landmark selection algorithm described in the previous section to improve the performance of simple schemes such as CBG and ShortestPing. In order to examine the ideal number of new landmarks needed we look at the median error of CBG and ShortestPing techniques as more landmarks are added to the landmark set. Figure 29 plots the localizaton error of CBG and ShortestPing schemes when new landmarks are added. We see that for both CBG and ShortestPing the knee of the curve is at around 90 landmarks – adding more landmarks does not substantially reduce localization errors. We also notice that ShortestPing has consistently much higher localization errors than CBG.

To examine this further we also look at the error distribution for 125 targets for different schemes in Figure 30. We see that adding 90 new landmarks almost halves the median error for CBG but is less effective at higher percentiles. The main reason for this is that CBG does not use path constraints and a passive landmark can

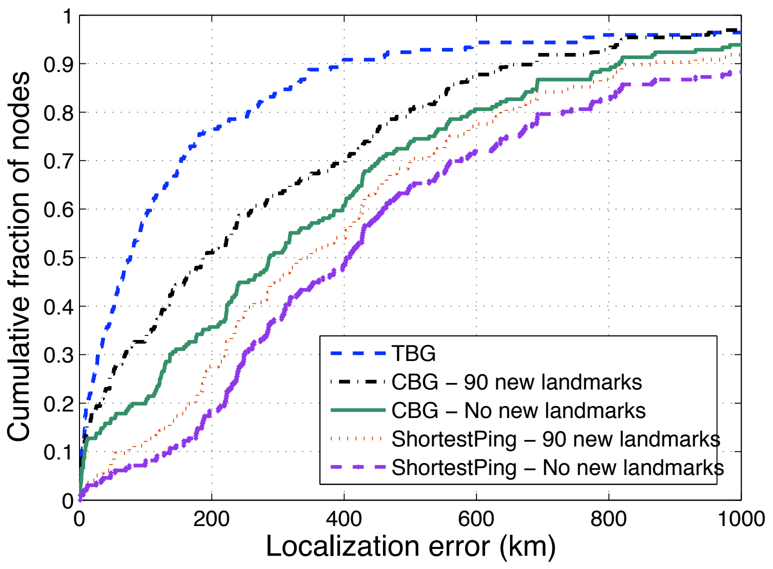

Figure 30: Distribution of localization errors for (i) TBG, (ii) CBG with 90 additional landmarks, (iii) CBG with no additional landmark, (iv) ShortestPing with 90 additional landmarks, and (v) ShortestPing with no additional landmarks. For CBG, median error improves by almost 100 kms while for ShortestPing median error improves by around 60 kms.

significantly affect a target only if it has end to end latency measurements to it; just being present on the path to the target does not suffice. For ShortestPing adding 90 new landmarks is less effective than for CBG but it still reduces the localization error as expected.

We believe that our scheme of adding passive landmarks would be more useful for schemes such as Octant which also make use of path information. We plan to evaluate these schemes as a part of future work.

3.7 Conclusions And Future Work

In this chapter we show that analyzing cause of errors in existing IP localization algorithms. The main cause of errors arise from a) incorrect measurements b) insufficient/incorrect constraints c) algorithmic limitations.

We show that techniques that use global optimization algorithms such as SDP perform quite well even if the target is far from landmarks. However for these algorithms the set of constraints needs to be carefully chosen. We show how the accuracy and speed of execution of these algorithms can be improved by choosing the right set of constraints. The use of SDP for real-time localization is limited because of computation time. Hence simple schemes are more useful.

To improve the accuracy of simple localization schemes we proposed a 2-tier algorithm which identifies a set of geographically distributed passive landmarks near most targets. Then simple schemes can be used in conjunction with these landmarks to improve localization accuracy.

As a part of future work we want to identify under constrained regions in the graph and identify what landmarks are needed to further improve the solution. We would also like to evaluate our algorithms on a larger dataset. In the future we plan to evaluate a number of other delay-based schemes such as Octant [104]. We would also like to try a testbed evaluation using Planetlab.

Chapter 4

Wireless Routing

4.1 Overview

Wireless mesh networks (WMNs) are multi-hop wireless networks that are gaining popularity as an alternative last-mile access technology besides DSL and cable modem. Nodes in WMNs are mostly stationary, and a small number of them are connected to the wired network serving as gateways. Since WMNs serve as access networks, maximizing utilization efficiency and throughput are the main goals in setting up such a network.

The challenges to this goal arise from the nature of the wireless channel. Radio signal propagation over a wireless link is affected by many factors such as attenuation caused by physical objects, signal strength, interference from other radio signal sources etc. As a result of the complex interactions among these mechanisms, a wireless link's signal quality and capacity tends to fluctuate significantly over time. Since the radio resource available to a WMN is limited, it is essential to maximize the utilization efficiency, even at the expense of increased control and system complexity.

In WMNs interference can significantly affect throughput achieved. Different packets of the same flow along adjacent links interfere with each other – this is called *intra-flow* interference. Packets of different flows within the radio range of each other also interfere and this is called *inter-flow* interference. Both these sources of interference affect the effective throughput of a WMN. One solution to limit interference is to make use of different non-interfering or orthogonal frequencies or channels available in wireless networks. For example 802.11b has 11 usable channels of which 3 are orthogonal. Authors in [82] show that using multiple channels can improve throughput by a factor of 6. Optimal assignment of channels in WMNs has been the focus of several previous works [6,84,98]. Even with optimal channel assignment flows on different channels can still experience some interference and hence can cause temporal fluctuations on the medium. Also channel quality is likely to deteriorate because of external interferences as well. Hence routing over the best path is orthogonal to quality of channel assignment. Routing algorithms still need to incorporate the effect of channel instability.

The other factor that affects performance on WMNs is loss rate. This specifically affects the performance of protocols such as TCP which use loss to indicate congestion. Several transport layer protocols have been proposed specifically for wireless networks which take into account the nature of the wirelss channel [50,83,85,93]. Although we have not examined the effect of channel fluctuations on the transport layer in this dissertation, we believe that the accurate measurement techniques developed and the metrics proposed in this chapter can be used to improve the performance of these transport protocols.

4.1.1 Routing In Wireless Mesh Networks

First-generation wireless network routing protocols such as AODV [81] and DSR [48] were primarily designed for mobile ad hoc networks, and their main design goals are to maintain network connectivity and to reduce the performance overhead associated with routing protocols. However, most WMN nodes are stationary and typically a few hops from gateway nodes connecting to the wired network. Therefore, the routing protocols for WMNs are more driven to maximize the overall network throughput by incorporating real-time link state and input load information, in a way similar to link-state routing protocols on wired networks.

Wireless Mesh routing protocols face the following design challenges:

- An optimal path over a wireless network depends on the quality of the underlying links. Hence, the algorithms have to use routing metrics which use cross layer information as opposed to generic routing metrics such as hop length.
- The wireless channel is very dynamic and the routing algorithms have to account for this behaviour. The reaction to a varying channel cannot be too frequent causing a lot of route flaps, at the same time not reacting to changes can cause sub-optimal routing.
- Since channel capacity is a valuable resource the routing algorithms should also be low overhead.

Popular wireless mesh routing protocols such as SRCR [14] and MCL [25] incorporate some of these design considerations. Both use wireless channel quality based routing metric for routing, and try to compute the best path based on the most current picture of the network. However, these protocols fail to incorporate the effects of temporal fluctuations in link quality. Our contribution to improving the performance of WMN routing protocols is a new routing protocol called the Channel Characteristic Aware Routing Protocol (CARP). In particular CARP features:

- An accurate wireless link capacity metric that is derived from per-packet transmission time measurements, and is able to capture the instantaneous *effective available capacity* of a wireless link. In addition it makes use of passive channel sensing to reduce measurement overhead without the need for additional monitoring cards.
- A stochastic wireless path capacity metric that uses a probability distribution to better model a WMN path's capacity profile over time, and hence is able to predict the effect of temporal fluctuations on path quality, and is less prone to path flapping.
- A multi-path routing algorithm that exploits path diversity to provide higher sustained throughput even in the presence of significant temporal fluctuations in wireless link quality.

In the rest of this chapter we look at different state of the art wireless mesh routing protocols, and then look at performance gains achieved with CARP.

4.2 Related Work

There is a large body of literature on channel quality aware routing protocols. We categorize these works based on their relevance to different protocol mechanisms incorporated into CARP.

4.2.1 Wireless Routing Metrics

Since wireless link quality is affected by many factors, empirical techniques are best suited to estimate it. The ETT link metric was proposed for the WCETT routing metric [26] which was designed for a multi channel network. WCETT of a *n* hop path is given by

$$WCETT = (1-\beta) * \sum_{i=1}^{n} ETT_i + \beta * max_{1 \leq j \leq k} X_j. \tag{1}$$

Here the X_j component represents channel diversity. If the system has a total of k channels

$$X_j = \sum_{Hop\ i\ is\ on\ channel\ j} ETT_i \quad 1 \leq j \leq k \tag{2}$$

and β is a parameter that controls the weight of each term. For a single channel network WCETT metric reduces to *ETTDelay* ($\beta = 0$) which is described in the next subsection. One main limitation of WCETT is that it does not accurately account for contention from other flows in the neighborhood and the suggested packet-pair technique to estimate link bandwidth, requires the use of explicit probe packets. The link quality metric proposed in CARP, called the Measured Transmission Time (MTT) metric, not only estimates the effective path bandwidth more accurately but also uses on-going transmissions to measure at a fine time scale. Note that MTT would work just as well even in a multichannel network since it would only account for those links which interfere in the same channel.

Several recent works [10,67,108,112] incorporate channel load or contention in the routing metric. The Medium Time Metric (MTM) [10] proposed by Awerbuch et al, uses the estimated end to end transmission time over a link. Estimated TT assumes that packets have fixed protocol overhead and channel access delay (back-off) is not included. The PARMA metric [112] uses passive channel sensing to measure the expected access delay of each packet and channel utilization to account for contention from other nodes. However, passive sensing with current wireless technology requires the use of an additional wireless card. The ICTT metric by Zhai et al. [108] is proven to be close to an optimal routing metric and uses the transmission time along the maximal clique of the path instead of the transmission time along the most limiting link. However, in real-world networks ICTT is hard to measure as cliques change over time. CARPs bandwidth based path metric, MTTBW is a close practical approximation to ICTT and does not require clique computations.

The weighted end to end delay metric (WEED) in [67] takes into account queueing and expected contention. It also accounts for intra flow interference by taking into account the number of 2 hop adjacent links in the same channel. This is similar to the way MTT accounts for intra-path interference. However to account for contention the authors use a theoretical estimate while MTT can correctly estimate the contention experienced on a link.

The goal of all the above works is to find the path with highest end-to-end throughput, but they do so indirectly by approximating the effective path capacity. We believe that the MTT metric is a practical metric that accurately estimates the effective link capacity and facilitates computing path bandwidth probability distributions to help make better routing decisions.

One recent work is a channel quality measurement framework called EAR [54]. This work looks at best techniques to measure channel quality, and shows that broadcast based probes to measure channel quality do not accurately reflect the behaviour of unicast traffic over the same links. Also, broadcast probes fail to account for unidirectionality [19] of wireless link quality. Both of these limitations are alleviated in our implementation since we use ongoing data transmissions and the metric is uni-directional.

4.2.2 Diversity And Multipath Routing

Multi-path routing has been used in several mobile wireless network protocols [65, 66] to increase reliability of data transfer in mobile ad-hoc networks. Multi-path routing can also be used in wireless networks to exploit channel diversity.

One approach to maximizing throughput is to leverage the broadcast nature of wireless transmissions. Specifically, when an intermediate receiver fails to receive a packet, one can leverage the fact that a neighbor of the intermediate receiver has successfully received the packet and could potentially forward it to the destination, thereby reducing retransmissions. ExOR is an opportunistic routing mechanism that makes use of the above intuition in making delayed forwarding decisions at the Routing layer [16]. Each relay node forwards packets only if it fails to overhear a higher priority node forwarding the same. However, it may not always be possible for neighbors of a node to overhear each other, and hence multiple copies of the same packet could be propagated in the network. ROMER [107], another routing protocol based on this framework leverages transient variations to select the highest throughput path using a credit based forwarding scheme. ROMER forms an opportunistic, forwarding mesh centered around the long-term stable, minimum-cost path which opportunistically expands or shrinks at runtime to exploit the highest-quality links.

Both ExOR and ROMER exploit the redundancy due to the broadcast nature of the medium, but do not explicitly leverage the multirate option at the physical layer. To overcome this limitation authors in [60] propose the Expected Anypath Transmission Time (EATT) metric which defines the expected transmission time from a node to a set of forwarding nodes on a given rate. To take advantage of this metric they also propose the Shortest Multirate Anypath First (SMAF) algorithm. Since all neighbors of a node are not accessible at all rates, each node maintains a set of neighbors associated with every transmission rate. For a given packet the neighbor set with the shortest EATT to the destination is chosen and the packet is transmitted at the corresponding rate. They prove that this algorithm is optimal and has the same complexity as the algorithm for the shortest single path.

One main limitation with these schemes is that they are not easily extensible to multi-channel networks. Multi-channel networks which provide much higher throughput [84], are usually designed such that neighbouring nodes are on different channels to reduce interference. In such a scenario the use of broadcast based multi-user diversity is limited. However, multi-path routing can be exploited even in multi-channel networks to increase end to end throughput and also make use of auto-rate mechanisms of modern cards. Multipath routing decisions such as number of paths and traffic distribution can be made dynamically based on existing channel conditions.

4.3 Experimental Platform

Evaluating wireless protocols is a challenging exercise in itself. Since modelling of wireless channel behaviour is very difficult, simulation based evaluation faces problems of experiment fidelity [20,22,40]. On the other hand deploying a full-scale wireless mesh network is challenging task. Also, once such a network is setup, few parameters of the network can be changed for evaluation. Hence, for evaluation of wireless protocols we would like to make use of wireless test-beds.

In recent years several wireless test-beds have been built such as ORBIT [3], Mobile Emulab [1], and MiNT-m [21,22]. The ORBIT testbed consists of fixed wireless nodes placed in a grid. However, this system does not give us the flexibility of varying the experiment topology. To change channel characteristics one has to introduce controlled noise into the test-bed. Computing the right amount of noise to provide realistic channel behaviour is a challenge in itself.

Mobile Emulab uses Acroname Garcia robots and the test-bed can be configured into different topologies. However, the nodes in the test-bed have to be manually recharged. Also, neither of these test-beds provide a convenient evaluation environment.

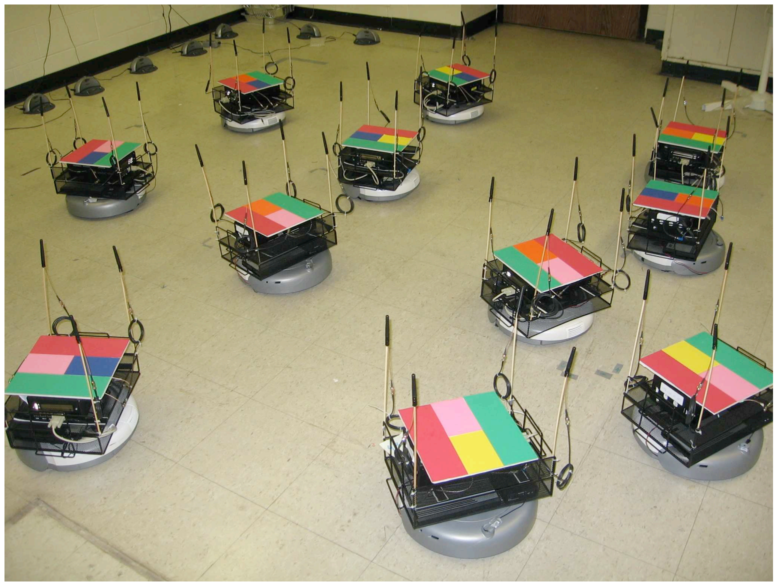

Figure 31: MiNT-m prototype with 12 mobile nodes and charging stations (top left corner of the image).

MiNT-mobile (MiNT-m), is a multi-hop mobile wireless network testbed that (1) supports experiment-by-experiment topology reconfigurability, (2) enables untethered node mobility, (3) can operate autonomously in a 24x7 fashion, (4) provides comprehensive monitoring and control of the network testbed, (5) supports hybrid ns-2 simulations, and (6) is deployable in a limited physical space using radio signal attenuation. Hence, we chose MiNT-m as our experimental platform.

4.3.1 MiNT-m Testbed

A mobile node in the MiNT-m testbed comprises of a wireless computing device and a mobile robot for physical movement. The wireless device is a RouterBoard (RB-230), that is a low-power battery-operated small form-factor computing board. Each Router- Board allows attaching 4 mini-PCI IEEE 802.11 a/b/g cards, that makes it possible to support multi-radio experiments. For the wireless cards used in the experiments, a radio signal attenuator is inserted between the wireless interface and its antenna to shrink the signal coverage area and thereby the physical space requirement.

The mobile robot is an inexpensive off-the-shelf robotic vacuum cleaner from iRobot, called Roomba. Necessary modifications to the Roomba are made to allow (i) the Roomba movements to be controlled from the wireless computing board mounted on it, and (ii) automatic recharging of a mobile node when the batteries drain out with suitable customization of the vendor supplied docking station, that has a self-homing feature to bring a Roomba back for recharging when in low-charge state. Moreover, each MiNT-m node uses a residual power estimation and a recharge scheduling algorithm to keep track of its battery status and determine the next recharge time

A PC equipped with multiple wireless network interfaces acts as the control server. All control traffic is transported over an IEEE 802.11g channel and thus does not interfere with IEEE 802.11a channels, which are used in actual experiments.

A vision-based positioning system is used to track the position of each mobile node in the testbed. The positioning system robustly tracks the nodes with zero false positives, and requires only commercial off-the-shelf webcams. The resulting node position estimates are used for monitoring and for planning trajectories for collision-free node movement. Figure 31 shows the MiNT-m testbed consisting of 12 nodes.

The key software components of MiNT-m are: (a) a control daemon (b) a node daemon component running on each node (c) the network monitor and control interface called MOVIE (Mint cOntrol and Visualization InterfacE). The software architecture of MiNT-m is shown in Figure 32 The control daemon collects position updates of the core testbed nodes from the tracking server, as well as, event traces from the nodes used in an experiment, which are used to update the visualization interface.

The MiNT-m test-bed provides a hybrid NS2 simulation environment, where the link and physical layers of NS2 are replaced with test-bed link and physical layers. This greatly simplifies testing new protocols, while maintaining experiment fidelity. The node daemon component issues commands to the hybrid simulator and collects traces from it to send to the control server. These traces are sent over the

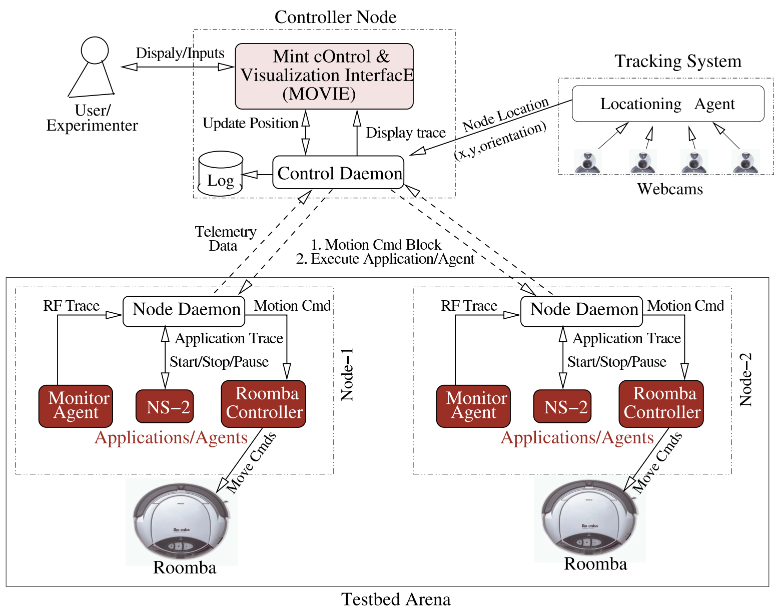

Figure 32: Overall MiNT architecture. The control daemon running on the control server collects inputs from the tracking server and the user, and controls the movement of mobile robots via the MOVIE interface for monitoring and control. Each testbed node corresponds to a Roomba robot and has a node daemon running on it, which communicates directly with the control daemon over a dedicated wireless control channel that is non-interfering with the channels used for the experiments. The core testbed nodes communicate with the peers using wireless NICs that are connected to low-gain antennas through radio signal attenuators. The vision-based tracking server periodically captures images of testbed nodes, and processes them to derive the location of each testbed node.

control channel to avoid interfering with the experiment. The node daemon is also responsible for controlling node movement.

The network/experiment management system designed for MiNT, called MOVIE (*Mint-m cOntrol and Visualization InterfacE*) coupled with hybrid NS2 provide an easy way to access and use the test-bed. Specifically, MOVIE provides real-time display of network traffic load distribution, pair-wise end-to-end routes, node/link liveliness, protocol-specific state variables, positions of individual nodes and inter-node signal-to-noise ratios. In addition, MOVIE allows users to control a simulation run dynamically, including pausing a simulation run at a user-specified breakpoint, inspecting its internal states and/or network conditions, modifying different simulation parameters, and resuming the run. It also supports a rollback

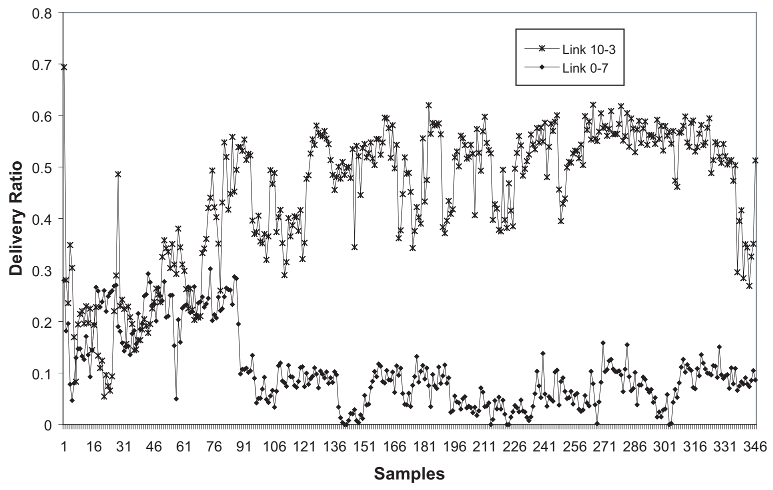

Figure 33: Wireless link loss variations for 2 test-bed links over a period of 1 hour

mechanism that allows one to go back to a previous state of a long-running simulation, and resume from there with a different set of simulation parameters. These features greatly aid in the development and evaluation of wireless protocols.

4.4 Channel Characteristic Aware Routing Protocol

CARP is a WMN routing protocol that leverages real-time radio channel quality information to maximize the sustained end-to-end throughput. It combines source routing with link state routing, and hence is similar in spirit to several recent WMN protocols such as Srcr [14] and MCL [25].

These protocols exploit the fact that most WMN nodes are stationary and typically a few hops from the wired network, and require every WMN node to compute a best path between itself and its corresponding gateway based on the WMN's link state information and certain optimization criteria. Although the topology of a WMN is static, the quality of its links could vary significantly over time. Therefore the "best" paths computed by these protocols tend to vary over time.

Figure 33 shows the substantial variation in the packet delivery ratio of two randomly chosen links on a 12-node Mint-m testbed over a 1-hour period.

Metric	Phy Rate	BER	Contention	Impl
ETX	No	Yes	No	Yes
ETT	Yes	Yes	No	Yes
PARMA	Yes	Yes	Yes	No
ICTT	Yes	Yes	Yes	No
Ideal	Yes	Yes	Yes	Yes

Table 2: This table shows the set of performance factors that existing wireless link capacity metrics incorporate, such as physical transmission rate (Phy Rate), the bit error rate (BER), the effect of contention from other nodes in the network and if a metric has a practical implementation.

These measurement were taken at night, when interference from external radio signal sources is minimum, and the testbed's topology remained fixed with minimal human movement that might affect radio channel conditions. Similar temporal fluctuations in wireless link quality have been observed by other researchers as well [4,19].

To reduce the performance impact of temporal fluctuation in wireless link quality, CARP was designed with the following three issues in mind:

- How to accurately estimate the effective available capacity of a wireless link in real time?
- How to properly combine per-link capacity measures into an end-to-end per-path capacity measure for route selection?
- How to choose an optimal number of paths for a given network flow and how to distribute the flow's traffic among the chosen paths?

In the following sections we discuss how CARP addresses each of these design issues in more detail.

4.4.1 Wireless Link Capacity Metric

Wireless link capacity metrics have been extensively studied [20,26,108,112]. To motivate the particular wirelss link capacity metric used in CARP, we first briefly review a representative subset of existing metrics.

4.4.1.1 Existing Link Capacity Metrics

One of the earliest works in this area was the Expected Transmission Count metric (ETX) [20], where the residual capacity of a link is modeled as the expected number of attempts needed to successfully transmit a packet over it. To improve over the ETX metric, Draves et al. [26] proposed the Expected Transmission Time (ETT) metric, which incorporates the effects of physical transmission rate and packet loss. The ETT metric of a wireless link is given by

$$ETT = ETX * (S/B). \tag{3}$$

where S is the packet size and B is the physical transmission rate.

One main limitation of ETT is that it does not account for contention from other nodes sharing the channel used by the wireless link in question. To address this problem, the authors suggest a packet-pair technique to estimate the effective value of B. However, this technique requires additional probe packets and therefore can only be applied sparsely. Using broadcast probes is not known to be very effective and unicast probes are needed to every neighbor at different channel rates.

Several recently proposed wireless link capacity metrics [10,108,112] further incorporated a radio channel's load and/or contention. The PARMA metric [112] uses passive channel sensing to measure the expected access delay and channel utilization for each packet to be transmitted, which requires an additional monitoring wireless network interface. The ICTT metric [108] uses packet transmission time as the wireless link capacity metric. However, no practical implementation of this metric exists.

There are several factor affecting a wireless link's effective residual capacity, which represents the additional load that the wireless link in question can further shoulder. The first factor is the physical transmission rate, i.e. the speed at which bits can be put on a wireless link. The second factor is interference from radio signal sources in the vicinity of the wireless link. This interference manifests itself as channel contention as well as bit errors. Therefore, an ideal wireless link capacity metric should explicitly incorporate the following factors into its definition: (i) the physical transmission rate (e.g., $1, 2, 6$ and 11 Mbps for IEEE 802.11b links), (ii) the bit error rate, and (iii) the contention among wireless nodes sharing the same radio channel. Table 2 summarizes the extent to which existing wireless link capacity metrics take into account these factors.

Analysis of wireless link quality measurements and existing wireless link capacity metrics suggests that

- No practical implementation of the ideal wireless link capacity metric exists.
- The time varying nature of wireless link quality demands constant real-time measurements of wireless link capacity.
- Given the relatively scarce radio resource, passive measurement is preferred to explicit probing.

4.4.1.2 Measured Transmission Time Metric

CARP approximates the ideal wireless link capacity metric using *end-to-end per-packet transmission time measurements*. The end-to-end transmission time (TT) of a packet over a wireless link is modeled as

$$TT = T_{access} + N_{retransmit} * \frac{S}{B} \tag{4}$$

where T_{access} represents the time a packet spends in exponential back-off or channel access, $N_{retransmit}$ is the number of times a packet is transmitted before receiving an ACK, S is the packet's size, and B is the wireless link's physical transmission rate. Because interference affects both contention and bit error rate, including these two factors in the metric implicitly accounts for interference.

Instead of computing TT, CARP uses a metric called the Measured Transmission Time (MTT) which approximates TT as follows. If there are already packets in the wireless interface, then the end-to-end transmission time of the i^{th} packet, TT_i, is equal to the difference between the arrival time of the i^{th} packet's ACK and that of the $i-1^{th}$ packet's ACK. That is,

$$TT_i = Ack_i - Ack_{i-1} \tag{5}$$

where, Ack_i is the time at which the ACK of the i^{th} packet arrives at the sender.

If no packet is currently in the wireless interface, then the end-to-end transmission time of the i^{th} packet, TT_i, is equal to the interval between the time when the i^{th} packet is DMAed into the wireless network interface and the time when the i^{th} packet's ACK is received. That is,

$$TT_i = Ack_i - Pkt_i \tag{6}$$

where Pkt_i is the time at which the ith packet is moved to the wireless interface. MTT_i is the normalized end-to-end per-packet transmission time, and is equal to $\frac{TT_i}{S_i}$, where S_i is the size of the ith packet in bytes.

Finally, a wireless link's MTT is a smoothed average of the normalized end-to-end per-packet transmission times for all packets traversing a wireless link. That is,

$$MTT = MTT_i * \beta + MTT * (1 - \beta) \tag{7}$$

where β is a smoothing constant that represents a tradeoff between responsiveness and stability.

In the above formulation, the per-packet queuing time is excluded since it has to do more with the input workload than with a wireless link's capacity. In fact, our experiences indicate that including the queuing time tends to introduce unnecessary fluctuations in the transmission time measurements as flows join and leave the network.

MTT is a uni-directional measurement in that the MTT for the $A \rightarrow B$ link is computed independently of that for the $B \rightarrow A$ link. Moreover, there is one MTT for each link between a pair of wireless interfaces rather than for each wireless interface.

To measure MTT we modified the Madfwifi driver for Atheros-based wireless LAN interfaces so that it generates an interrupt when a MAC-layer ACK packet is received. In addition, we leveraged the 64-bit timestamp counter in these WLAN interfaces to take high-resolution timing measurements. We verified the validity of this scheme by running a single UDP flow over a wireless link, measuring its application-level throughput, and comparing it with the bandwidth estimate derived from MTT. The results show that the MTT-based link capacity estimate closely matches the measured application-level throughput.

Several of the previous works have suggested using the TT metric as a way to estimate a wireless link's capacity. For example, the authors of WCETT metric [26] proposed per-packet transmission time as a possible ideal metric. Also, MTM [10], PARMA [112] and ICTT [108] are approximations to TT. However, no real implementations of the TT metric on commodity WLAN NIC have been demonstrated. One contribution of this work is a practical implementation of the TT metric that is validated on a real-world wireless testbed.

Several previous works on cross-layer wireless transport protocols have used packet transmission characteristics for rate control. EXACT [49] is a transport protocol which uses a rate based congestion control scheme. EXACT routers measure the available bandwidth as the inverse of per-packet MAC contention and transmission time. Each router then runs a proportional max-min fair bandwidth sharing algorithm to divide this measured bandwidth among the flows passing through it. Because of the difficulty in getting per packet MAC contention and transmission time values this protocol could only be evaluated using simulations.

Similarly LRTP [85], a stateful transport protocol for WMNs also uses transmission time value to do explicit rate control. In their implementation they use a separate monitoring channel to measure packet transmission times. This not only reduces the available NICs per node but can also be inaccurate when there are errors in monitoring. Our modification to the madwifi driver can be leveraged by both these works to allow a realistic evaluation of these protocols.

4.4.2 Wireless Path Capacity Metric

To compute a routing path, one needs to combine the residual capacity measures of constituent wireless links, such as MTT values, into a wireless path capacity metric. There are two possibilities. If a wireless link's residual capacity is measured in terms of packet delay, the right way to combine them into a path capacity measure is to sum them up and the result corresponds to the path's end-to-end delay. If a wireless link's residual capacity is measured in terms of available bandwidth, the right way to combine them into a path capacity measure is to take the minimum of them and the result corresponds to the path's end-to-end bandwidth. Given the MTT of a wireless link, its available bandwidth can be approximated by the inverse of MTT.

As longer paths consume more network resources, most existing wireless network routing protocols take the summing approach to favor paths with a fewer number of hops and thereby prevent a flow from consuming too much network resource.

Although summing the per-link packet transmission times minimizes a flow's end-to-end delay, it does not necessarily maximize a flow's throughput. For example, suppose there are two paths $P1$ and $P2$, whose bottleneck links' available bandwidths are $B1$ and $B2$, respectively, and whose end-to-end delays are $D1$ and $D2$, respectively, where $B2 > B1$ and $D1 < D2$. In this case, $P2$ would provide better throughput, even if $P1$ offers a smaller end-to-end delay.

From the above analysis, CARP adopts the following two capacity metrics for a wireless path based on the MTT values of its constituent links:

$$MTTDelay_p = \sum_{l \in p} MTT_l * I_l^p \tag{8}$$

$$MTTBW_p = \frac{1}{MAX_{l \in p}(MTT_l * I_l^p)} \tag{9}$$

where $MTTDelay_p$ represents the expected end-to-end delay for a new flow's packets traversing the path p, whereas $MTTBW_p$ represents the bandwidth available to a new flow. $MTTBW_p$ does not take into consideration a wireless path's length, and thus is likely to favor paths consisting of higher-bandwidth links even if they require more hops.

I_l^p represents the degrading impact on the MTT of link l due to the flow's packets crossing l's neighboring links on path p (intra-flow interference). It represents

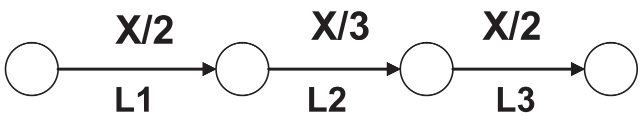

Figure 34: An example wireless network to illustrate the effect of intra-path interference. Each links capacity is X, but a flow going over all three links will only get $X/3$.

the degree of self interference experienced by a flow traversing link l on path p. For non-edge links on the wireless path p, I_l^p is set to the minimum of 3 and the number of adjacent links in p in the same channel. For edge links (links in the beginning and the end of the path) I_l^p is set to minimum of 2 and the number of adjacent links in p in the same channel. This assumes that intra-flow interference extends to a maximum of one neighboring hop on each side of a link.

To illustrate how I_l^p accounts for the effect of self interference, let's consider a wireless path that consists of 3 links, L1, L2 and L3, each of which has a link capacity X as shown in figure 34. Initially, the path is completely idle, so the initial capacity of the path is $min(\frac{X}{2}, \frac{X}{3}, \frac{X}{2}) = \frac{X}{3}$. Assume the first flow traversing this path consumes $\frac{X}{6}$ of the path's initial capacity $(\frac{X}{3})$. When the second flow comes along, the measured capacities of L1, L2 and L3 are $\frac{2X}{3}$, $\frac{X}{2}$ and $\frac{2X}{3}$, respectively, and the available path capacity for the second flow is thus $min(\frac{X}{3}, \frac{X}{6}, \frac{X}{3}) = \frac{X}{6}$, which is exactly the difference between the path's initial capacity and the first flow's consumption.

Although the I_l^p heuristic is not always accurate, it represents the best approximation that does not require network-specific physical interference measurements. Another alternative to accurately measure intra-flow interference would be to use a candidate set of paths (top 2 or 3 with the lowest cost or highest throughput). Send test flows along these paths and then use $MTTDelay$ and $MTTBW$ measurements for these test-flows to select the best path. This clearly has much higher overhead but can correctly account for intra-flow interference. However in practice we see that our heuristic works quite well in distinguishing better paths.

It should be noted that neither of these two path capacity metrics account for temporal fluctuations in wireless channel quality.

4.4.3 Accommodating Temporal Fluctuation

Temporal fluctuations in wireless link quality, as shown in Figure 33, could have a significant performance impact. Since path bandwidths vary significantly over time the best routing decision made at time T may no longer be optimal at time $T+\delta$. In this subsection we present two techniques to incorporate wireless link quality fluctuation into packet routing decisions.

4.4.3.1 Modeling Path Capacity As A Probability Distribution

For many transport protocols that are TCP-friendly [1], the variance of a wireless path's capacity is as important as its mean because these protocols react very poorly to sudden increase in packet loss. Therefore, in many cases, a wireless path with a smaller mean and variance may well be a better choice from the standpoint of transport-layer performance. To demonstrate this point, we performed a ns2 simulation for four 2-link paths. The first path goes through two 11Mbps links, each with a loss rate of 50%. The second path goes through a 11Mbps link and a 5.5Mbps, each with a loss rate of 30%. The third path goes through two 5.5Mbps links, each with a loss rate of 50%. The fourth path goes through two 1Mbps links, each with a loss rate of 0%. We measured the throughput of a TCP connection running over each path, and at the same time measured the per-link MTTs, from which we can derive the mean and variance of each path's capacity. The results of this experiment are shown in Table 3. Although the first path has the highest mean path capacity (1.42Mbps), its TCP throughput is the lowest because its variance is the largest (50%). In contrast, the fourth path has the lowest mean path capacity (0.33Mbps), but its effective TCP throughput is actually the highest, because its variance is zero.

Recognizing that both the mean and variance of a wireless path's capacity play an important role, CARP chooses to treat the capacity of a wireless path p as a probability density function F_p, rather than a scalar value, as is the case with

[1]TCP-friendly protocols are those whose packet sending rate is at most some constant over the square root of the packet loss rate. A TCP-friendly protocol would not have an unfair advantage over TCP when sharing the same channel.

Mean	Variance (Percent)	TCP Throughput
1.42Mbps	0.71Mbps (50%)	0.019Mbps
0.72Mbps	0.24Mbps (34%)	0.13Mbps
0.475Mbps	0.1Mbps (22%)	0.20Mbps
0.33Mbps	0Mbps (0%)	0.33Mbps

Table 3: The mean and variance of the capacity of four simulated two-link paths, and the throughput of a TCP connection running over each of these paths.

existing wireless routing protocols. To account for path capacity fluctuations in routing we first compute the cumulative distribution function (CDF) of each path's capacity. The CDF of a path p ($F_p(B)$) is

$$F_p(B) = Pr(x \leq B) \tag{10}$$

$$G_p(B) = 1 - F_p(B) = Pr(x > B) \tag{11}$$

where $G_p(B)$ is the Complimentary CDF (CCDF) of p's capacity and is more convenient than $F_p(B)$ because it provides a lower bound on a path's capacity.

The CCDF of a path's capacity can be derived from the CCDFs of its constituent links, if we apply Equation 9 to combine link capacity measures into a path capacity measure.

Assume a N-hop path p consists of the following set of links, $l_1, l_2 ..., l_N$, each of whose capacity CCDF is denoted as G_{l_i}. Let X_p be the random variable representing p's capacity, and X_{l_i} be the random variable representing the capacity of p's ith link. Then the CCDF of p's capacity can be expressed as

$$\begin{aligned} F_p(B) = \quad & Pr(X_p \leq B) = 1 - Pr(X_p \leq B) \\ = \quad & 1 - Pr(X_{l_1} > B) * Pr(X_{l_2} > B) * ..Pr(X_{l_N} > B) \\ = \quad & 1 - G_{l_1}(B) * G_{l_2}(B) * ..G_{l_N}(B) \end{aligned} \tag{12}$$

and its complement is thus

$$G_p(B) = 1 - F_p(B) = G_{l_1}(B) * G_{l_2}(B) * ..G_{l_N}(B) \tag{13}$$

In CARP, each individual wireless link's capacity CCDF (G_{l_i}) is derived from a histogram of measurements over a sliding window.

The next question is how to choose the best path among a set of candidate paths based on the CDFs of their path capacities. CARP takes the following approach: It computes the maximal capacity that each path is able to guarantee with a probability threshold T, and chooses the path with the highest guarantee. For example, if T is set to 0.9, CARP calculates the bandwidth B_T for the ith path such that $G_i(B_T) = 0.9$, and chooses the path with the highest B_T. T is an empirical constant that is set to 0.8 (20th percentile) in our experiments.

Mathematically, the best path p_{best} is the one that satisfies the following:

$$p_{best} = arg\,max_{p_i}(G_{p_i}^{-1}(0.8)) \tag{14}$$

The above formulation makes no assumptions about the type or form of the distribution functions, and thus could be applied to arbitrary network paths whose capacity CDF is computable. Intuitively, with $T = 0.8$,

CARP is essentially looking for the path with the best 20th percentile bandwidth, which is expected to be close to a path's mean ca- pacity if its path capacity is stable, but may deviate significantly from its mean if the quality fluctuation of the path's underlying links is substantial.

CARP assumes a WMN which is connected to the wired backbone through a set of gateway nodes, and every WMN node is connected to the wired backbone through a particular gateway node. Every T (current set to 1) seconds, each WMN node computes an MTT value for each wireless link in which its wireless network interfaces are involved, and sends these MTT values to its associated gateway node. Each gateway node uses the past N (currently set to 30) MTT samples it received for every link in the network, and computes the link's capacity CDF.

Given a pair of source and destination nodes, the gateway computes the top K paths [27] between them by combining per-link capacity CDFs into a per-path capacity CDF using Equation 13, and then determining the desirability of each candidate path using Equation 14. In the rest of this chapter we will refer to this CDF-based routing algorithm as *MTTProb*, which is summarized in Algorithm 2.

Algorithm 2 MTTPROB $(LinkCache, Pr, N_S, N_P)$

```
Input: $LinkCache$, $Pr$ (Probability Parameter), $N_S$ (Samples), $N_P$ (Number of Paths)
Result: $BestPath$
 1: $LinkCache$.ComputePerLinkCDF ($N_S$)
 2: $TopPaths = LinkCache$.GetTopPaths ($MTTBW, N_P$)
 3: $BestPath$ = -1
 4: $BestBw$ = -1
 5: for $Path_i$ in $TopPaths$ do
 6:     $CCDF_i = Path_i$.ComputeCCDF()
 7:     $B_i = CCDF_i$.GetBandwithWithProbability ($Pr$)
 8:     if $B_i > BestBw$ then
 9:         $BestBw = B_i$
10:         $BestPath = Path_i$
11:     end if
12: end for
13: return $BestPath$
```

4.4.3.2 Exploiting Diversity Through Multi-path Routing

In the above subsection, we assume that between a sender and a receiver, some paths are inherently better than others after properly accounting for their link quality fluctuation behaviors. However, in practice, there are many cases in which over the long term, there is no clear winner among the candidate paths being considered but at any point in time, some paths are better than others. In the case that the rate at which better paths change is faster than the rate at which the routing protocol can afford to react, the best course of action is to use multiple paths to carry a flow in the hope that they can complement one another at the times when some of them suffer drastic quality degradation.

There are two key decisions in the design of a multi-path routing protocol: (i) which subset of paths to use, and (ii) how to distribute a flow's traffic among the paths.

CARP uses the following algorithm to address these two issues:

Algorithm 3 MTTMUL $(LinkCache, Pr, N_S, N_P, B_r)$

Input: *LinkCache*, *Pr* (Probability Parameter), N_S (Samples), N_P (Number of Paths), B_r (Bandwidth requirement)

Result: $BestPathSet, Weights$ or $BestPath$

```
1: $BestBw = 0$
2: $BestAssignment = [\ ]$
3: $Unchanged = 0$
4: $THRESH = 100$ (Settling Threshold)
5: $DisjointPaths = LinkCache$.GetTopDisjointPaths ($N_P$)
6: while TRUE do
7:     $A$ = ComputeRandomAssignment ($DisjointPaths$)
8:     for $Path_i$ in $DisjointPaths$ do
9:         $LinkCache$.ComputePerLinkCDF ($N_S$,$A$)
10:        $CCDF_i = Path_i$.ComputeCCDF()
11:    end for
12:    $CCDF_M = Convolve(CCDF, A)$
13:    $B_i = CCDF_M$.GetBandwithWithProbability ($Pr$)
14:    if $B_i > BestBw$ then
15:        $BestBw = B_i$
16:        $BestAssignment = A$
17:        $Unchanged = 0$
18:    else if $Unchanged > THRESH$ then
19:        break
20:    else
21:        $Unchanged += 1$
22:    end if
23: end while
24: $BestSinglePath = MTTProb(LinkCache, Pr, N_S, N_P)$
25: if $BestSinglePath.BestBw > BestBw$ then
26:     return $BestSinglePath$
27: else
28:     return $DisjointPaths, BestAssignment$
29: end if
```

1. Given a sender and a receiver, use the $MTTBW$ metric to compute a set of node-disjoint paths D_p, which is initially empty. At each iteration the best path from the remaining network is added to D_p and the links corresponding

to that path are removed from further consideration. This is repeated K times so that D_p eventually has at most K node-disjoint candidate paths.

2. Compute the capacity CDF for each of the K paths found in Step 1. Let the CDF of the ith path be $F_i()$, its offered load be a_i, the end-to-end throughput of the ith path be b_i, and the cumulative distribution function for b_i be $T_i()$. Then

$$T_i(X) = \begin{cases} F_i(X) & if \;\; X \leq a_i, \\ 1 & if \;\; X > a_i \end{cases} \tag{15}$$

   The end-to-end throughput of a path cannot exceed its offered load.

3. The total throughput of using all K paths, B, is equal to $\sum b_i$, and is a convolution of the $T_i()$ associated with these paths. However, in practice, there is inter-path interference among these K paths. We incorporate this effect by modifying the value of I_l^p in Equations 8 and 9 to include both intra-path and inter-path interference.

4. To search for the best a_i's that maximizes $1 - T^{-1}(0.8)$, we start by partitioning the bandwidth requirement of the user flow among the K candidate paths in a weighted fashion, where the weight assigned to the ith path is proportional to $1 - T_i^{-1}(0.8)$. Because the search space is large, we sample a number of weight assignments randomly varying the initial weights, and settle down to the weight assignment that maximizes $1 - T^{-1}(0.8)$. For each candidate weight assignment, repeat Step 2 and 3.

5. Compare the 20th percentile bandwidth of the best multi-path routing choice with that of the best single-path routing choice obtained from $MTTProb$, and choose the better of the two. This way CARP uses multi-path routing only when it is advantageous to do so.

To distribute a flow's traffic load among the constituent paths of a multi-path route, the current CARP prototype uses a random sampling approach to simplify the implementation complexity. For more optimal results, more sophisticated techniques such as gradient descent could be used.

The combination of these two techniques result in the $MTTMul$ algorithm, which is summarized in Algorithm 3.

4.4.4 Summary

In summary, CARP applies a series of optimizations to advance the state of art in wireless routing protocols. First, it applies a per-packet transmission time measurement approach to accurately estimate the effective residual capacity of a wireless link. Second, to accommodate wireless link quality fluctuations, CARP uses a probability distribution rather than a single scalar value to model a wireless path's capacity, and determine the desirability of a path based on its probability distribution function. This approach allows CARP to clearly discern between paths with different degrees of link quality fluctuation. Finally, to exploit diversity among multiple candidate paths for which there is no clear winner and which may be able complement one another, it extends the probability distribution model to multi-path routing and uses random sampling to solve the traffic distribution problem in multi-path routing. While CARP is more involved than previous wireless routing protocols, the fact that it is designed for routing traffic between pairs of WMN nodes rather than individual flows allows it to amortize the route computation cost over multiple flows over a period of time.

4.5 Evaluation Methodology

We evaluated the performance of different routing schemes on the MiNT-m testbed Each physical MiNT-m node maps to a wireless node in each experiment and runs an instance of the hybrid ns2 simulator. In the following experiments, we assumed every WMN node has only one NIC and they all operate in the same channel.

We used a 3x4 grid topology for evaluation, and set the WLAN cards to operate in the 802.11a mode. All the nodes operated on the same channel, but physical transmission rates and loss rates of different links vary. The mesh network emulated on the Mint-m testbed forms a logical tree rooted at a gateway node, Once every second each MiNT-m node sends a link state update to the gateway node. In each run, the gateway first computes a path between itself and every testbed node based on link quality information of the past 30 seconds under a background traffic load, using the following routing schemes: *ETTDelay*, *MTTBW*, *MTTDelay*, *MTTProb* and *MTTMul*; then a set of test connections runs for 20 seconds using the computed paths and their throughputs are measured, analyzed and reported.

Additionally, for large simulated topologies, we took a trace-driven approach. More specifically, we collected link- layer packet transmission traces from three different types of links on the Mint-m testbed. The three types of links corre- spond to good ($<$ 5% loss rate), medium (5-40% loss rate), and poor ($>$ 40% loss rate). For each link type, we collected multiple packet transmission traces. and played them inside the ns2 simulator. As a result, the packet loss behavior of a simulated link is similar to a testbed link in terms of both average packet loss rate and temporal burstiness of packet losses. However, this approach does not improve the ns2 interference model.

4.5.1 MTT Metric Implementation

We modified the Madwifi driver to implement the MTT metric. The driver was modified to request an interrupt for every successful packet transmission. As described previously if there are multiple packets in the device queue then the transmission time of the packet is the difference between successive acknowledgements. If there is a single packet in the queue then the transmission time is the difference between when the packet was queued to when the interrupt was received.

The driver maintains the following stats for each outgoing link – average MTT, physical transmission rate and retransmission count. The driver exposes these measurements to user-level programs through Linux's `/proc` interface. The emulated link layer of the hybrid ns2 simulator reads these values and maintains the current ETT, and MTT values for each wireless link in which it is involved.

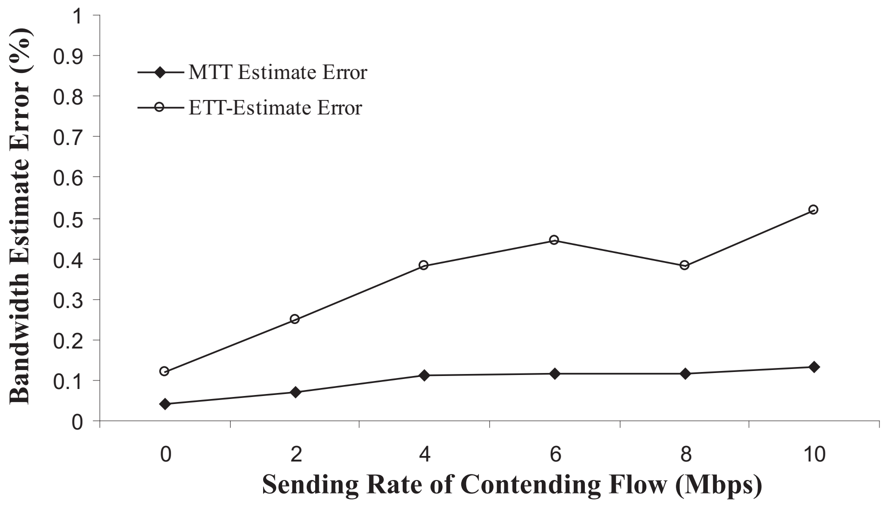

Figure 35: The percentage estimation errors of MTT and ETT for a UDP connection running over a one-hop wireless link with a sending rate of 6Mbps when there is a contending UDP connection that varies its sending rate from 0Mbps to 10Mbps. ETT's estimation error increases with the background load, while MTTs estimation error remains almost constant.

4.6 Results And Analysis

The goal of this performance evaluation study is to empirically quantify the throughput gain of CARP over known wireless routing algorithms, and to characterize the performance contribution of its component mechanisms. The main performance metric used is connection throughput expressed in bits/sec. We compare the proposed MTT metric with the ETT metric [26] where the ETT of a path is the sum of each constituent links ETT and is referred to as $ETTDelay$.

In [26], the authors use a packet-pair based technique to determine B, which requires explicit probing of each link with a packet pair. This can be expensive especially in a loaded channel and would not accurately capture short term channel variations or auto transmission rate adjustment. Hence, for our evaluation we have set the value of B to the physical transmission rate reported by the driver.

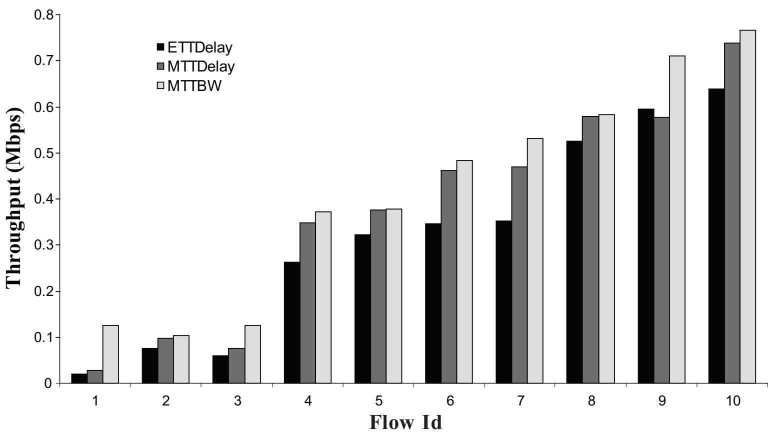

Figure 36: End-to-end throughput measurements of 10 best flows on the MiNT-m testbed for ETTDelay, MTTDelay and MTTBW schemes in the presence of a background flow with a sending rate of 2Mbps. MTTDelay and MTTBW always outperform ETTDelay. MTTBW performs the same or better than MTTDelay in all cases.

4.6.1 Effectiveness Of MTT

To evaluate the effectiveness of MTT as a wireless link capacity metric, we compare the application-level throughput of a UDP connection under different background loads with the predictions from MTT and ETT metrics.

We set up a UDP connection that sends packets at 6Mbps over an IEEE 802.11a link that is configured to physically transmit at 6Mbps. This ensures that the link under consideration is saturating the wireless channel even in the absence of interference. The background load is in the form of another UDP connection over a neighboring IEEE 802.11a link that operates in the same channel but at the physical transmission rate of 24Mbps.

Figure 35 shows the estimation errors of MTT and ETT as the background UDP connection's sending rate varies from 0 to 10 Mbps. As the background load of the radio channel increases, the throughput estimation error of ETT increases since it does not consider contention and interference from the background source. In contrast, MTT's throughput estimate error remains largely the same even in the

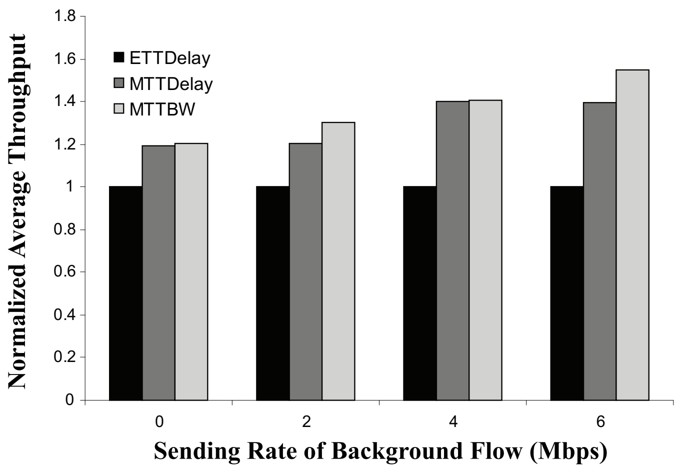

Figure 37: Normalized average throughput of the 20 flows on the MiNT-m testbed for ETTDelay, MTTDelay and MTTBW when the background traffic load is varied from 0 Mbps to 6 Mbps. MTTDelay and MTTBW are more effective than ETTDelay because ETTDelay does not explicitly take into account contending traffic. MTTBW shows 20-60% improvement over ETTDelay and upto 15% over MTTDelay

BG Load	ETTDelay	MTTDelay	MTTBW
0Mb	2.72	2.17	2.25
2Mb	2.70	2.23	2.27
4Mb	2.68	2.34	2.34
6Mb	2.72	2.44	2.43

Table 4: The average path length of 20 flows (from figure 37), as the sending rate of the background flow increases. The average path length for ETTDelay does not change as it does not adapt to the increasing background load, whereas the average path length for MTTBW and MTTDelay increases slowly with the increasing background load as flows are routed through more hops to stay away from congested links.

presence of large background loads. Although MTT accounts for all delay components associated with a packet's transmission, it does not account for all performance overheads and therefore still exhibits a 5-15% estimation error.

4.6.1.1 Comparison Among ETTDelay, MTTDelay And MTTBW

We next investigate if taking the inverse of the MTT of a path's bottleneck link (MTTBW) is a better path capacity metric than taking the sum of per-link ETTs or

MTTs of a path (ETTDelay, MTTDelay).

To compare them, we compare the paths selected based on these three path capacity metrics. Because our testbed is a small one, we focus on 20 randomly chosen node pairs, which share a common node, the gateway. In each run, for the first 30 seconds the other node of each node pair sends link updates to the gateway. At the end of the 30 seconds, the other node of each node pair sends a route request to the gateway, which computes a route and sends back a reply, and the non-gateway node starts a test flow after receiving the gateway's reply.

To ensure that we have sufficient interfering traffic we use a 1-hop flow in the center of the testbed as background traffic. This background flow has a sending rate of 2Mbps.

The average throughputs of the 10 best flows under the three routing metrics are shown in Figure 36. In almost all cases, MTTDelay and MTTBW metrics outperform ETTDelay, thus further validating that MTT is a better capacity metric than ETT. In addition, MTTBW performs equally well or better than MTTDelay. This suggests that as far as connection throughput is concerned the "bottleneck bandwidth" approach is more effective than the "sum of delay" approach, and that paths with higher bandwidth could incur longer delay in many cases.

To assess the impact of the background load's volume, we repeated the same experiment but varying the sending rate of the background traffic from 0Mbps to 6Mbps. The choice of background traffic here is to show the effect of increasing background load on the path capacity metrics. One problem with high background traffic load is that it increases the bit error rate and in turn causes route request packets to be dropped. In a real wireless network, this is not a major problem because lost packets are retransmitted. However, this problem seriously hampers experiment repeatability. Therefore we cap the background traffic load to 6Mbps, which is not very high but still can saturate a 802.11a channel.

Figure 37 shows the average throughput of the 20 flows for each path capacity metric. The throughput values are normalized with respect to ETTDelay. To ensure fair comparison, throughput measurements for different path capacity metrics for a particular node pair and background load were run back to back.

Path	Mean	Var	Y=0.7	Y=0.8	Y=0.9	Throughput
P_{11}	0.679	0.134	0.65	0.60	0.50	0.695
P_{12}	0.682	0.260	0.65	0.55	0.45	0.470
P_{21}	1.10	0.310	1.0	1.0	1.00	0.829
P_{22}	1.31	0.453	1.2	1.1	0.95	0.983

Table 5: The mean, variance and path capacity with probability Y and the actual measured throughput of four paths over the MiNT-m testbed. P_{11} and P_{12} are for one node pair, and P_{21} and P_{12} are for another node pair.

As expected, the performance gains of MTT-based metrics over ETTDelay increase with the background load, because the difference between ETT and MTT is more likely to manifest itself under heavier loads.

To give insight into the underlying dynamics, we show the average path length of the 20 flows under the three path capacity metrics in Table 4. The average path length for ETTDelay remains unchanged as the background load increases, as this path capacity metric is less sensitive to a particular constituent link's load. In contrast, the average path lengths for MTTDelay and MTTBW increase with background load as they steer away from congested links by using longer paths.

Note that the paths ETTDelay chose are generally longer than those chosen by MTTDelay and MTTBW in this setup, because ETTDelay used the physical transmission rate for the value of B (Equation 3), and thus was more likely to select paths that go through nodes with higher physical transmission rates, even if the links experience lower throughput due to interference.

4.6.2 Path Capacity As A Probability Distribution

To compare MTTProb, which approximates a wireless path's capacity as a probability distribution, with MTTBW and MTTDelay, which represent a wireless path's average capacity, we selected two node pairs in the MiNT-m testbed, and for each pair picked two alternative paths. Let P_{11} and P_{12} be the two alternative paths for the first node pair, and P_{21} and P_{22} for the second node pair.

For each of these paths, we collected the link quality characteristics for 30

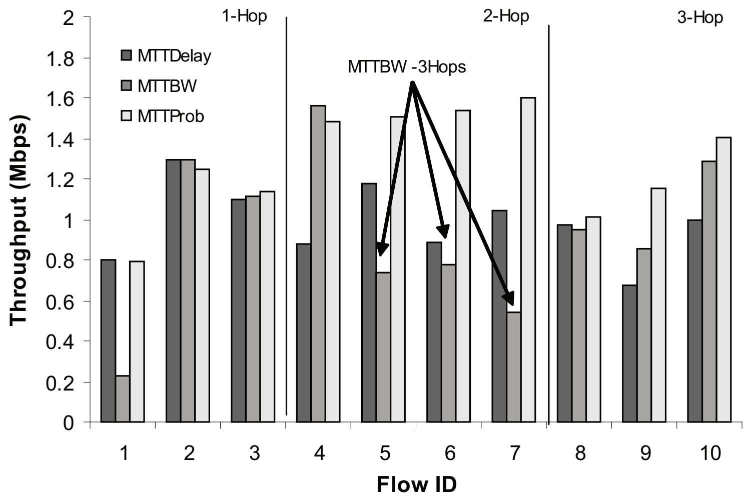

Figure 38: Throughput of 10 TCP Flows on the MiNT-m testbed for MTTProb (T=0.8), MTTDelay and MTTBW. Flows are classified by the number of hops in the path chosen by MTTProb. MTTDelay outperforms MTTBW for Flow 1,5,6,7 because MTTBW chooses a longer path for those flows.

seconds and used the resulting information to derive the mean, variance and path's capacity with particular probability. Then we measured the actual throughput of each path and used it as the ground truth for comparing routing metrics. The results of these measurements are shown in Table 5, where T is set to $0.7, 0.8$ and 0.9. The *mean* capacity alone is not a reliable metric, because P_{12} is worse than P_{11} in terms of actual throughput even though P_{12} has a higher mean than P_{11}. In this case, the main cause is the high variance. $T = 0.8$ seems to be the best choice in this case as routing decisions based on it are consistent with those derived from actual measured throughputs.

In general, higher T results in a more conservative path capacity estimate whereas lower T tends to over-estimate the actual path capacity. How T can be adapted to a given transport protocol such as TCP according to the impact of link quality fluctuation on its throughput is an interesting research topic that is worth further exploration. Because the path capacity CDFs are computed from histograms which contains discrete values, using a percentile path capacity estimate could often lead to a tie among multiple candidate paths. Currently we resolve ties by choosing

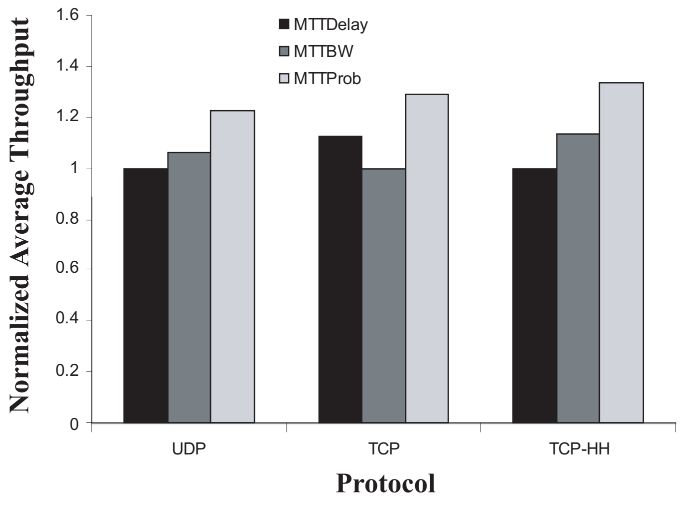

Figure 39: Normalized average throughput of 20 flows on the MiNT-m testbed under three routing metrics, MTTProb, MTTDelay and MTTBW and three transport protocols, UDP, TCP and TCP-HH. MTTProb improves over MTTDelay by 20% for UDP, by 28% for TCP and 33% for TCP-HH

the path with the smaller fractional variance, i.e., $\frac{variance}{mean}$.

In general, MTTProb could make a wrong routing decision in two cases. Firstly if the path follows the exact bandwidth distribution as that observed based on measurements prior to routing. In this case the long term average bandwidth of the path would equal the mean rather than the 20th percentile (Y=0.8). Secondly, MTTProb may err if the path quality actually improves to that of a peak after the routing decision has been made. However, in our experiments we have seen that both these cases are rare, and taking a conservative estimate to identify the best path is actually beneficial.

A probability distribution-based path capacity estimate is particularly useful for transport protocols such as TCP that are sensitive to fluctuations in path quality.

To evaluate this we again used the 3x4 grid topology. However, due to TCPs sensitivity to loss rate the testbed topology was carefully setup to have lower bit error rates 10% for most links and at worst 30%. Setting up such a topology requires positioning nodes such that link quality to most neighbors is good, as opposed to the previous experiment where ensuring that the network is connected was sufficient.

We also had to choose the background traffic carefully. A high rate background flow can increase the loss rate of the channel and significantly affect TCP performance. (The performance limitations of TCP on wireless are well known [83, 85]). This would also make repeatability of experiments across different routing schemes impossible. Therefore we chose background flows such that, though the rate of each flow was low, there were multiple of them to cause sufficient interference. The background traffic consisted of 3 UDP flows placed at different points in the grid, each sending traffic at the rate of 200Kbps.

We ran 20 TCP flows over 20 randomly selected node pairs on the MiNT-m testbed. Here again, one of each node pair plays the role of a gateway. The experiments were repeated 3 times for each scheme and there were 180 runs in total. The measured throughputs of 10 of these 20 TCP flows under MTTProb (T=0.8), MTTDelay, MTTBW are shown in Figure 38. Except for two flows (2 and 4), MTTProb performs better than the other two, because it prefers stable paths with lower variance. Also, the average throughput for MTTBW is lower than that of MTTDelay. This is because MTTBW sometimes chooses longer paths with higher mean capacity and a larger number of hops tends to increase the packet loss probability. This adversely affects TCP because it backs off unnecessarily whenever it detects consecutive packet losses.

Because TCP is not the most effective transport protocol for multi-hop wireless networks, we repeated the same experiment using a more appropriate transport protocol, hop-by-hop TCP (TCP-HH). Figure 39 shows the normalized average throughputs of the 20 TCP flows under three routing metrics, MTTProb (T=0.8), MTTDelay and MTTBW, and three transport protocols, TCP, UDP and TCP-HH. In all cases, MTTProb outperforms MTTDelay and MTTBW. This is more so because the CDF based scheme is indeed able to better approximate a path's capacity than a mean value, which is less reliable when there is substantial link quality fluctuation. For UDP and TCP-HH, MTTBW performs better than MTTDelay because the adverse effect of packet losses on connection throughput is largely removed.

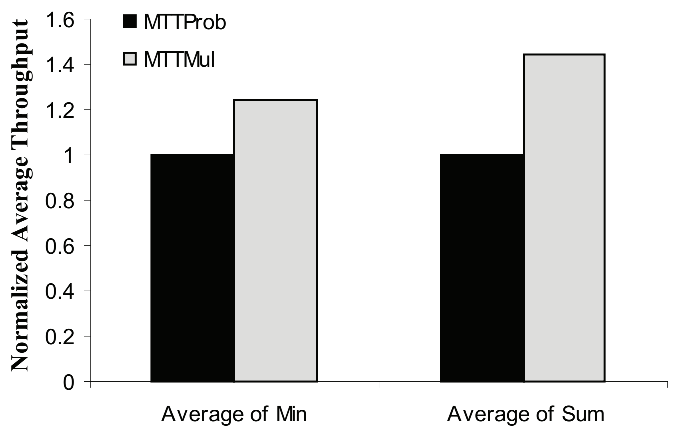

Figure 40: Normalized averages of the minimum and sum of each flow pair's throughputs for 20 flow pairs running on the MiNT-m testbed under MTTMul and MTTProb schemes. MTTMul improves minimum by 24% and the sum by 44%

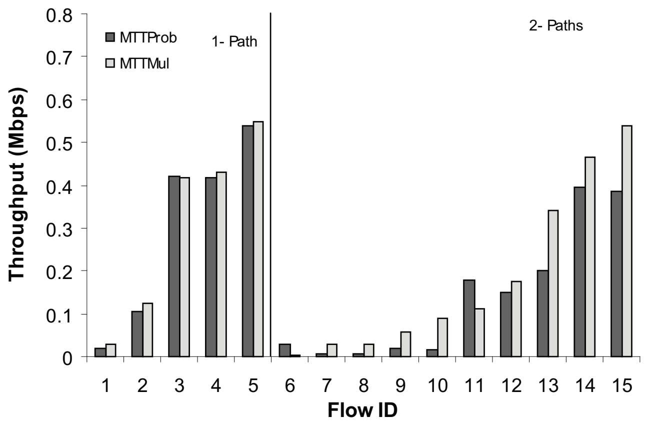

Figure 41: Throughputs of 15 simultaneous UDP flows on a 5x8 (40 Node) simulated network under MTTMul and MTTProb schemes. MTTMul reduces to MTTProb for some flows where a single good path exists. When MTTMul chose two-path routes, it outperforms MTTProb in all cases except for Flow 7 and 11.

4.6.3 Multi-path Metric

Compared with MTTProb, MTTMul further improves the robustness of a wireless routing decision by leveraging diversity in multiple paths. However, using multiple paths has it cost, in the form of increased self interference and adverse impact of packet reordering. If the best path between a node pair exhibits a consistently

Scheme	**Average (Mbps)**	**Aggregate (Mbps)**	**Fairness Index**
MTTMul	0.225	3.4	0.308
MTTProb	0.192	2.91	0.197

Table 6: Average throughput, aggregate throughput and fairness index of MTTMul and MTTProb for the 15 flows used in the simulation study whose results are shown in Figure 41.

high quality, using multiple paths to route traffic between that node pair would only degrade the effective throughput. Therefore, in its current design, MTTMul only chooses a multi-path route if and only if such a route's estimated capacity is higher than that of the single best path found using MTTProb.

To compare MTTMul with MTTProb, we randomly chose 40 node pairs in the 3x4 grid, set up a UDP flow between each node pair, and ran two flows at a time. In each run, the two simultaneous flows are routed either both using MTTMul or MTTProb, and we measured the minimum (MIN) and the sum (SUM) of these two flows' throughputs. The average results for the measurements of 20 such runs are shown in Figure 40. MTTMul outperforms MTTProb in both MIN and SUM measurements. MTTMul does better in MIN throughput because using multiple paths provides an effective hedge against sudden decrease in link quality especially for poorly connected node pairs. MTTMul also improves the aggregate throughput of the two test flows as it utilizes more network resources simultaneously.

To provide further understanding of how multi-path routing helps, we performed a series of trace-driven simulations. The network topology used in the simulations was a 5x8 (40-node) grid. The quality distribution of links in the simulated network was 10% good links, 46% medium links and 44% poor links, which corresponded to the link characteristics observed in the 3x4 MiNT-m testbed. Two of the 40 nodes were designated as gateways, and flows were set up between non-gateway nodes to one of the two gateways. We simulated 15 simultaneous UDP flows on the test network and measured their throughputs. The results are shown in Figure 41, organized according to the number of paths used in the final route selected by MTTMul. When MTTMul chooses a single-path route, it is always the same as

MTTProb's choice. For this topology MTTMul never uses more than two paths as the increase in self interference prevents it from choosing more than two paths.

When MTTMul chooses two-path routes, it outperforms MTTProb in all cases except for Flow 7 and 11. This is to be expected because the 15 flows were sharing the entire network resource and the performance improvement for some flows may come at the cost of degraded performance of others. However, because MTTMul utilizes the network resources more efficiently and is able to exploit path diversity the aggregate throughput and the fairness index computed as $(\sum_i X_i)^2/n * \sum_i {X_i}^2$, where X_i is the throughput of i^{th} flow, of these 15 flows are all higher under MTTMul than under MTTProb, as shown in Table 6.

4.6.4 Lessons

We summarize some of the lessons we learned from this performance evaluation study. Firstly, evaluating a wireless protocol in a real wireless network is very hard, especially when we need to trade off between fidelity and repeatability. Second, because of the inherent temporal fluctuation of wireless link quality, it is difficult to ensure that different schemes being compared are run under exactly the same radio channel conditions. Third, TCP's performance over wireless networks can easily be an order of magnitude worse than UDP, and is very dependent on the channel condition. Hence, any TCP-related evaluation requires the network topology to be carefully setup in order to limit the impact of channel loss and variance.

4.7 Conclusion

In this dissertation we take the approach of first defining the essential dimensions in the design space of WMN routing protocols, and then comparing the design alternatives incorporated into specific routing protocols in each dimension. We believe this methodology provides more useful and meaningful information than a protocol-by-protocol comparison. Because of the large number of design choice combinations, this work represents the first step toward a complete comparative study of throughput-maximizing routing protocols for wireless mesh networks. Nonetheless, important lessons have been learned and they are

- MTT is a more accurate wireless link capacity metric because it incorporates all performance factors including contention delay.
- Taking the bottleneck link's residual capacity is a more effective way to estimate a wireless path's capacity than summing up the delays of the path's constituent links.
- Using a probability distribution to model a wireless path's capacity is more robust in the presence of temporal fluctuation of wireless link quality, than using a single scalar value.
- Multi-path routing can improve the throughput robustness of wireless flows only if it is used when it is beneficial to do so.

We also conclude that routing in emerging networks such as Wireless mesh networks can be completely designed with specific performance goals in mind. Also incorporating good measurement metrics in the protocol is an important aspect of improving performance in highly dynamic environments.

Bibliography

[1] Emulab Tutorial - Mobile Wireless Networking. http://www.emulab.net/tutorial/mobilewireless.php3.

[2] PlanetLab. http://www.planet-lab.org.

[3] Whynet. http://pcl.cs.ucla.edu/projects/whynet/.

[4] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 121 – 132, 2004.

[5] Akamai, Inc. `http://www.akamai.com`.

[6] M. Alicherry, R. Bhatia, and L. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proceedings of the 11th annual international conference on Mobile computing and networking (MOBICOM)*, pages 58 – 72, 2005.

[7] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 131 – 145, Oct. 2001.

[8] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. Anderson, and P. N. Belhumeur. A Theory of Network Localization. In *IEEE Transactions on Mobile Computing*, pages 1663 – 1678, Dec 2006.

[9] J. Aspnes, D. Goldenberg, and Y. R. Yang. On the computational complexity of sensor network localization. In *The First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 32 – 44, 2004.

[10] B. Awerbuch, D. Holmer, and H. Rubens. The medium time metric: High throughput rate selection in multi-rate ad hoc wireless networks. In *To appear in Kluwer Mobile Networks and Applications (MONET) Journal Special Issue on Internet Wireless Access: 802.11 and beyond*, pages 201 – 205, 2006.

[11] M. Badoiu, E. D. Demaine, M. T. Hajiaghayi, and P. Indyk. Low-dimensional embedding with extra information. In *Proceedings of the twentieth annual symposium on Computational geometry (SCG)*, pages 320 – 329, 2004.

[12] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of IEEE INFOCOM*, volume 2, pages 775 – 784, 2000.

[13] A. R. Berg and T. Jordán. A proof of Connelly's conjecture on 3-connected generic cycles. *Journal of Combinatorial Theory, Series B*, 88:77 – 97, May 2003.

[14] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proceedings of the 11th annual international conference on Mobile computing and networking (MOBICOM)*, pages 31 – 42, September 2005.

[15] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the third international symposium on Information processing in sensor networks (IPSN)*, pages 46 – 54, 2004.

[16] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. In *Proceedings of ACM SIGCOMM Computer Communication Review*, pages 69 – 74, 2004.

[17] CAIDA. `http://as-rank.caida.org/`.

[18] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *Proceedings of IEEE INFOCOM*, pages 1724 – 1751, 2000.

[19] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC)*, pages 414 – 425, 2005.

[20] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MOBICOM)*, pages 419 – 434, September 2003.

[21] P. De, A. Raniwala, R. Krishnan, K. Tatavarthi, J. Modi, N. Syed, S. Sharma, and T. Chiueh. Mint-m: an autonomous mobile wireless experimentation platform. In *Mobisys*, pages 124 – 137, 2006.

[22] P. De, A. Raniwala, S. Sharma, and T. Chiueh. Mint: A miniaturized network testbed for mobile wireless research. In *Proceedings of IEEE INFOCOM*, pages 2731 – 2742, March 2005.

[23] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 43 – 56, 2007.

[24] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions on Networking*, 12(6):963 – 977, 2004.

[25] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. In *Proceedings of ACM SIGCOMM*, pages 133 – 144, September 2004.

[26] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *10th annual international conference on Mobile computing and networking*, pages 114 – 128, September 2004.

[27] D. Eppstein. Finding the k shortest paths. In *Society for Industrial and Applied Mathematics Vol. 28, No. 2*, pages 712 – 716, Jul 1998.

[28] T. Eren, D. Goldenberg, W. Whitley, Y. R. Yang, A. S. Morse, B. D. Anderson, and P. N. Belhumeur. Rigidity, Computation, and Randomization of Network Localization. In *Proceedings of IEEE INFOCOM*, pages 2673 – 2684, 2004.

[29] B. Eriksson, P. Barford, J. Sommers, and R. Nowak. A learning-based approach for ip geolocation. In *Passive and Active Measurement*, volume 32, pages 171 – 180. Springer Berlin / Heidelberg, 2010.

[30] N. Feamster, J. Winick, and J. Rexford. A model of bgp routing for network engineering. In *ACM SIGMETRICS*, pages 331 – 342, June 2004.

[31] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs. Locating Internet routing instabilities. *Proceedings of ACM SIGCOMM*, pages 205 – 218, 2004.

[32] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *Proceedings of Symposium on Networked System Design and Implementation (NSDI)*, pages 239 – 252, 2004.

[33] M. J. Freedman, K. Lakshminarayanan, and D. Mazieres. OASIS: Anycast for any service. In *Proceedings of Symposium on Networked System Design and Implementation (NSDI)*, pages 129 – 142, 2006.

[34] L. Gao and J. Rexford. Stable internet routing without global coordination. In *Transactions on Networking*, pages 681 – 692, Dec. 2001.

[35] J. E. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*. Graduate Studies in Math., AMS, 1993.

[36] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-based geolocation of internet hosts. *IEEE/ACM Transactions in Networking*, 14(6):1219 – 1232, 2006.

[37] S. Guha, R. Murty, and E. G. Sirer. Sextant: a unified node and event localization framework using non-convex constraints. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC)*, pages 205 – 216, 2005.

[38] P. Gupta and P. R. Kumar. The capacity of wireless networks. In *IEEE Transactions on Information Theory*, pages 388 – 404, 2000.

[39] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker. On routing asymmetry in the Internet. In *Autonomic Networks Symposium in Globecom*, pages 6 – 13, 2005.

[40] J. Heidemann, N. Bulusu, and J. Elson. Effects of Detail in Wireless Network Simulation. In *Proc. of the SCS Multiconference on Distributed Simulation*, pages 1 – 10, January 2001.

[41] B. Hendrickson. Conditions for unique graph realizations. *SIAM J. Comput.*, 21(1):65 – 84, 1992.

[42] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning Systems: Theory and Practice*. Springer, 5 edition, 2001.

[43] C. Huang, A. Wang, J. Li, and K. W. Ross. Measuring and evaluating large-scale CDNs. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 15 – 29, 2008.

[44] ISC. `www.isc.org/index.pl?/ops/ds`.

[45] D. J. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *Journal of Computational Physics*, 137(2):346 – 365, 1997.

[46] V. Jacobson. Pathchar. `ftp://ftp.ee.lbl.gov/pathchar`.

[47] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MOBICOM)*, pages 66 – 80, Sep 2003.

[48] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad-hoc wireless networks. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 158 – 163. Kluwer Academic Publishers, 1996.

[49] N. V. K. Chen, K. Nahrstedt. The utility of explicit rate-based flow control in mobile ad hoc networks. In *IEEE Wireless Communications and Networking Conference, (WCNC)*, pages 1921 – 1926, 2004.

[50] Karthikeyan Sundaresan and Vaidyanathan Anantharaman and Hung-Yun Hsieh and Raghupathy Sivakumar. ATP: a reliable transport protocol for ad-hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC)*, pages 64 – 75, 2003.

[51] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards ip geolocation using delay and topology measurements. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 71 – 84, New York, NY, USA, 2006. ACM.

[52] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. van Wesep, T. Anderson, and A. Krishnamurthy. Reverse traceroute. In *Proceedings of Symposium on Networked System Design and Implementation (NSDI)*, pages 219 – 234, 2010.

[53] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson. Studying black holes in the Internet with Hubble. In *Proceedings of Symposium on Networked System Design and Implementation (NSDI)*, pages 247 – 262, 2008.

[54] K. Kim and K. G. Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking (MOBICOM)*, pages 38 – 49, 2006.

[55] R. Krishnan, H. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize cdn performance. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 190 – 201, 2009.

[56] G. Laman. On graphs and rigidity of plane skeletal structures. In *Journal of Engineering Mathematics*, pages 4:331 – 340, 1970.

[57] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. E. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. N. Schilit. Place lab: Device positioning using radio beacons in the wild. In *Proc. of Pervasive*, pages 116–133, 2005.

[58] R. Laufer, H. Dubois-Ferriere, and L. Kleinrock. Multirate anypath routing in wireless mesh networks. In *Proceedings of IEEE INFOCOM*, pages 37 – 45, 2009.

[59] S. Lederer, Y. Wang, and J. Gao. Connectivity-based localization of large scale sensor networks with complex shape. In *Proceedings of IEEE INFOCOM*, pages 789 – 797, May 2008.

[60] A. Lee and I. Streinu. Pebble Game Algorithms and Sparse Graphs. In *Third European Conference on Combinatorics - Graph Theory and Applications, Third European Conference on Combinatorics*, pages 1425 – 1437, April 2008.

[61] A. Lee, I. Streinu, and L. Theran. Graded Sparse Graphs and Matroids. In *Journal of Universal Computer Science*, volume 13, pages 1671 – 1679, 2007.

[62] A. Lee, I. Streinu, and L. Theran. The slider-pinning problem. In *Proceedings 19th Canadian Conference on Computational Geometry*, Aug 2007.

[63] S. J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hocnetworks. In *IEEE International Conference on Communication (ICC)*, pages 3201 – 3205, April 2001.

[64] R. Leung, J. Liu, E. Poon, A. C. Chan, and B. Li. MP-DSR: A qos-aware multi-path dynamic source routing protocol for wireless ad-hoc networks. In *26th Annual IEEE International Conference on Local Computer Networks (LCN),*, pages 132 – 141, April 2001.

[65] H. Li, Y. Cheng, C. Zhou, and W. Zhuang. Minimizing end-to-end delay: A novel routing metric for multi-radio wireless mesh networks. In *Proceedings of IEEE INFOCOM*, pages 46 – 54, 2009.

[66] LimeLight Networks, Inc. `http://www.limelightnetworks.com`.

[67] K. Lougheed and Y. Rekhter. Border Gateway Protocol. In *Request for Comments RFC-1105, Internet Engineering Task Force (IETF)*, June 1989.

[68] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 367 – 380, 2006.

[69] H. V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: Path prediction for peer-to-peer applications. In *NSDI*, pages 137 – 152, 2009.

[70] R. Mahajan. *Practical and Efficient Internet Routing with Competing Interests*. PhD thesis, University of Washington, 2005.

[71] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet path diagnosis. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 106 – 119, 2003.

[72] R. Mahajan, M. Zhang, L. Poole, and V. Pai. Uncovering performance differences among backbone ISPs with Netdiff. In *Proceedings of Symposium on Networked System Design and Implementation (NSDI)*, pages 205–218, 2008.

[73] D. Meyer. RouteViews. `http://www.routeviews.org`.

[74] M. Muuss. Ping. `http://ftp.arl.mil/\~mike/ping.html`.

[75] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings of IEEE INFOCOM*, pages 170 – 179, June 2002.

[76] V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. In *Proceedings of ACM SIGCOMM*, pages 173 – 185, New York, NY, USA, 2001. ACM.

[77] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM ToN*, 5(5):601 – 615, Oct. 1997.

[78] R. Percacci and A. Vespignani. Scale-free behavior of the internet global performance. In *The European Physical Journal B - Condensed Matter and Complex Systems*, volume 32, pages 411 – 414. Springer Berlin / Heidelberg, 2003.

[79] C. Perkins and E. M. Royer. Ad hoc on-demand distance vector (aodv) routing. In *Internet-Draft, draft-ietf-manet-aodv-04.txt*, Oct 1999.

[80] A. Raniwala and T. Chiueh. Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network. In *Proceedings of IEEE INFOCOM*, pages 2223 – 2234, 2005.

[81] A. Raniwala, P. De, S. Sharma, R. Krishnan, and T. Chiueh. End-to-end flow fairness over ieee 802.11-based wireless mesh networks. In *Proceedings of IEEE INFOCOM*, pages 2361 – 2365, May 2007.

[82] A. Raniwala, K. Gopalan, and T. Chiueh. Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. In *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, pages 50 – 65, April 2004.

[83] A. Raniwala, S. Sharma, P. De, R. Krishnan, and T. Chiueh. Evaluation of a stateful transport protocol for multi-channel wireless mesh networks. In *IEEE Workshop on Quality of Service (IWQoS)*, pages 74 – 82, June 2007.

[84] T. S. Rappaport. *Wireless Communications: Principles and Practice*. IEEE Press, Piscataway, NJ, 1996.

[85] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. In *Proc. of WCNC*, pages 1664–1669, Mar 2005.

[86] Y. Rekhter and T. Li. Border Gateway Protocol 4 (BGP-4). In *Request for Comments RFC-1771, Internet Engineering Task Force (IETF)*, March 1995.

[87] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: a case for informed Internet routing and transport. *IEEE Micro*, 19(1):50 – 59, 1999.

[88] A. Savvides, C. C. Han, and M. B. Strivastava. Dynamic fine-grained localization in *ad-hoc* networks of sensors. In *Proceedings of the 7th annual international conference on Mobile computing and networking (MOBICOM)*, pages 166 – 179, Rome, Italy, July 2001. ACM Press.

[89] J. B. Saxe. Embeddability of weighted graphs in *k*-space is strongly NP-hard. In *Proceedings of the 17th Allerton Conference in Communications, Control and Computing*, pages 480 – 489, 1979.

[90] I. J. Schoenberg. Remarks to maurice fréchet's article "sur la définition axiomatique d'une classe d'espace distanciés vectoriellement applicable sur l'espace de hilbert". *Ann. Math.*, 36(3):724 – 732, 1935.

[91] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. Wtcp: A reliable transport protocol for wireless wide-area networks. In *Proceedings of the 5th annual international conference on Mobile computing and networking (MOBICOM)*, pages 231 – 241, 1999.

[92] A. M. So and Y. Ye. Theory of semidefinite programming for sensor network localization. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 405 – 414, 2005.

[93] J. L. Sobrinho. Network routing with path vector protocols: Theory and applications. In *Proceedings of ACM SIGCOMM*, pages 113 –124, Aug. 2003.

[94] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *Proceedings of ACM SIGCOMM*, pages 93 – 106, 2003.

[95] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM ToN*, pages 2 – 16, 2004.

[96] A. P. Subramanian, H. Gupta, S. R. Das, and J. Cao. Minimum interference channel assignment in multi-radio wireless mesh networks. In *IEEE Transactions on Mobile Computing (TMC)*, pages 1459 – 1473, December 2008.

[97] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The impact of routing policy on internet paths. In *Proceedings of IEEE INFOCOM*, pages 736 – 742, 2001.

[98] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering what-if deployment and configuration questions with WISE. In *Proceedings of ACM SIGCOMM*, pages 99 – 110, 2008.

[99] K. Varadhan, R. Govindan, and D. Estrin. Persisten route oscillations in inter-domain routing. In *Computer Networks*, pages 32(1):1–16, 2000.

[100] L. Wang, K. Park, R. Pang, V. S. Pai, and L. L. Peterson. Reliability and security in the CoDeeN content distribution network. In *USENIX Annual Technical Conference*, pages 171–184, 2004.

[101] Y. Wang, S. Lederer, and J. Gao. Connectivity-based sensor network localization with incremental delaunay refinement method. In *Proceedings of IEEE INFOCOM*, pages 2401 – 2409, April 2009.

[102] B. Wong, I. Stoyanov, and E. G. Sirer". Octant: A comprehensive framework for the localization of internet hosts. In *Proceedings of Symposium on Networked System Design and Implementation (NSDI)*, pages 313 – 326, 2007.

[103] J. Wu, Z. M. Mao, J. Rexford, and J. Wang. Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network. In *Proceedings of Symposium on Networked System Design and Implementation (NSDI)*, pages 1 – 14, 2005.

[104] G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19 – 22, 1938.

[105] Y. Yuan, H. Yang, S. Wong, S. Lu, and W. Arbaugh. Romer: Resilient opportunistic mesh routing for wireless mesh networks. In *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, September 2005.

[106] H. Zhai and Y. Fang. Impact of routing metrics on path capacity in multirate and mutihop wireless ad hoc networks. In *Proceedings of International Conference on Network Protocols (International Conference on Network Protocols (ICNP))*, pages 86 – 95, November 2006.

[107] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In

Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI), pages 167 – 182, 2004.

[108] Y. Zhang, Z. M. Mao, and M. Zhang. Effective diagnosis of routing disruptions from end systems. In *Proceedings of Symposium on Networked System Design and Implementation (NSDI)*, pages 219 – 232, 2008.

[109] Y. Zhang, V. Paxson, and S. Shenker. The stationarity of Internet path properties: Routing, loss, and throughput. Technical report, ACIRI, 2000.

[110] S. Zhao, Z. Wu, A. Acharya, and D. Raychaudhuri. Parma: a phy/mac aware routing metric for ad-hoc wireless networks with multi-rate radios. In *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, (WoWMoM)*, pages 286 – 292, June 2005.