2015

A NETWORK PATH ADVISING SERVICE

Xiongqi Wu
University of Kentucky, snowcc@gmail.com

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Xiongqi Wu, Student

Dr. James Griffioen, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

A NETWORK PATH ADVISING SERVICE

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
Xiongqi Wu

Lexington, Kentucky

Director: Dr. James Griffioen, Professor of Computer Science

Lexington, Kentucky

2015

ABSTRACT OF DISSERTATION

A NETWORK PATH ADVISING SERVICE

A common feature of emerging future Internet architectures is the ability for applications to select the path, or paths, their packets take between a source and destination. Unlike the current Internet architecture where routing protocols find a single (best) path between a source and destination, future Internet routing protocols will present applications with a set of paths and allow them to select the most appropriate path. Although this enables applications to be actively involved in the selection of the paths their packets travel, the huge number of potential paths and the need to know the current network conditions of each of the proposed paths will make it virtually impossible for applications to select the best set of paths, or just the best path.

To tackle this problem, we introduce a new *Network Path Advising Service (NPAS)* that helps future applications choose network paths. Given a set of possible paths, the NPAS service helps applications select appropriate paths based on both recent path measurements and end-to-end feedback collected from other applications. We describe the NPAS service abstraction, API calls, and a distributed architecture that achieves scalability by determining the most important things to monitor based on actual usage. By analyzing existing traffic patterns, we will demonstrate it is feasible for NPAS to monitor only a few nodes and links and yet be able to offer advice about the most important paths used by a high percentage of traffic. Finally, we describe a prototype implementation of the NPAS components as well as a simulation model used to evaluate the NPAS architecture.

KEYWORDS: Future Internet, Path Selection, Scalable Monitoring

Xiongqi Wu

March 26, 2015

A NETWORK PATH ADVISING SERVICE

By

Xiongqi Wu

Dr. James Griffioen

Director of Dissertation

Dr. Miroslaw Truszczynski

Director of Graduate Studies

March 26, 2015

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my advisor Dr. James Griffioen, who has been a great mentor for me. I would like to thank him for encouraging my research and for helping me becoming a better computer scientist. His knowledge and enthusiasm in computer networks guided me towards the correct path for my Ph.D study. His advice on how to tackle a problem has been priceless. I appreciate him for sending me to conference where I met a lot of people and learned different ideas. Without his supervision and constant help this dissertation would not have been possible.

I would like to thank Dr. Zongming Fei, Dr. Miroslaw Truszczynski, and Dr. James Lumpp for being my committee members and for their help in my Ph.D study. I would also like to thank Dr. Aaron Cramer for being my outside examiner.

In addition, I would like to thank Hussamuddin Nasir, Lowell Pike, William Marvel for their assistance in maintaining lab machines, setting up testbeds, and fixing some technical problems for my projects.

I want to give a special thanks to my parents. Their endless love and support for me was what sustained me thus far. Their encouragement gave me a great confidence to pursue my doctoral study in Computer Science.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In today's Internet, communication between two end hosts occurs over a path selected by the Internet's routing protocols. Network routing in the Internet is (for the most part) based on distributed algorithms in which routers collectively identify the best path for packets to travel between a source and destination. For example, routing tables are maintained by various routing protocols, such as BGP or OSPF. The *Border Gateway Protocol (BGP)* [1] exchanges routing information between *autonomous systems (ASes)*, and makes routing decision based on connectivity, network policies, and rules enforced by network administrators. On the other hand, the *Open Shortest Path First (OSPF)* [2] routing protocol typically operates within a single AS, providing interior routes for an AS. OSPF picks the shortest path as the routing path. Neither BGP nor OSPF offers applications control over the routing paths. In other words, applications have little choice but to use the single path selected by the routing service, regardless of whether the selected path meets the needs of the application.

There are a variety of reasons why applications want to control the paths their packets take. These reasons are often based on performance requirements such as bandwidth, loss rate, latency, or reliability, but are requirements that will vary from application to application. For example, a video server streaming high definition video may choose to take the route that has high bandwidth and low jitter, while a

game application might want a low delay path.

To address this problem, many emerging network architectures include the ability for applications to select the path (or set of paths) they want to use from a list of potential network paths. For example, overlay networks often give applications the ability to choose among a set of routes for a given source and destination pair. In a programmable virtual network, such as GENI [3], the network can be programmed to route different types of traffic along different paths. Future Internet architectures, such as NEBULA [4], also offer multiple choices of routing paths. In addition, the source routing approaches, such as ICING [5], XIA [6], and POMO [7], enable applications to specify their own routing paths. Even the IPv4 and IPv6 protocols have defined a way to support loose source routing although it is often disabled. Therefore, we envision that in future networks, applications will have more control over the paths their packets traverse across the network. For example, Internet service providers (ISPs) can deploy *software defined networks (SDNs)* [8] to give applications control over the routing paths. While these approaches hold great potential, they require that new types of routing services and protocols be developed that can identify and return a list of paths to applications – a list from which the application will then choose.

Since future routing protocols are likely to provide applications with multiple choices of routes, helping applications choose suitable routing paths has become an important problem. If an application must pick its own routing paths, it will need dynamic, up to date information to help it decide what routing paths it should take and how to make the most effective use of those paths. Even if only a small number of paths are provided by the future routing service between a source and destination, there could be a huge number of paths network-wide. Monitoring all paths applications could possibly choose will not scale. In addition, some links on these paths cannot be monitored because ISPs may not allow monitoring on certain

links.

To choose routing paths that offer the best or most appropriate service for an application, this thesis proposes a new *Network Path Advising Service (NPAS)*. Note that NPAS is neither a routing service (which finds connectivity) nor a traditional monitoring service (that captures detailed measurements of node and link performance). Instead, the goal of the NPAS service is to take paths found by a routing service as well as measurement information captured by NPAS to develop estimates of *end-to-end path performance* and then make recommendations about which paths to use and how to use them based on the application's unique requirements.

1.1 Desired features of NPAS

To help applications determine suitable paths, NPAS must rate routing paths based on the application's requirements. Providing accurate path ratings requires NPAS to collect dynamic (i.e., rapidly changing) information about the links and routers that comprise a path such as the available bandwidth, queuing delays, jitter, loss rates, etc. There are two ways that NPAS can obtain dynamic path information:

1. Collect information from the network (i.e., *monitor* the Internet)

2. Collect information from users (e.g., ask applications running on end systems to report feedback about their experience).

1.1.1 Rating Paths

Future routing services will need to find and make available a *set of paths* that could be used by the application. Given the fact that current routing protocols already face scalability challenges finding a single path, one can be assured that finding a *set of paths* will present even greater scalability challenges. Given a massive number of possible paths between a source and destination, simply finding some subset of

those paths will require significant computation. Note that finding connectivity is application independent and applications with different requirements could receive a similar set of routing paths. Even if the routing service could find all possible paths, it will not be capable of rating them because the "performance" of the path is constantly changing and it is infeasible for the routing service to monitor the performance of all the paths given the measurement and processing overhead needed to do so. For example, in the current Internet, there are tens of thousands of ASes. Even if a single path is used for applications between each pair of ASes, it will introduce an unacceptable amount of overhead to monitor thousands of millions of paths.

To help applications select the best path for their needs, the NPAS system must rate/rank paths based on the application's needs. Rating the paths found by the routing service is a hard problem because:

- The ranking depends on the application's requirements for the path. The ranking can only be done when NPAS knows which application needs a path (and the characteristics of the path that is needed).

- The characteristics of the path are constantly changing and must be continuously collected to make informed decisions. Since fast collection of path information consumes more network resources than infrequent collection of path information, NPAS must carefully decide what links/nodes it should monitor in order to scale.

- The path information needs to be distributed to "local NPAS decision making processes" because sending all path ranking requests from all clients in the Internet to a central server will not scale (note that path ranking requests occur more frequently than, say, Google search requests). In addition, information needs to be distributed at a rate based on the request patterns to help reduce the traffic overhead.

- NPAS must deal with partial path information. NPAS may have some information along a path, but not information for all links. Partial information could be beneficial even though it is incomplete, providing information needed to select a path that may potentially meet the application's need.

- Even if NPAS has all the latest up-to-date information, computing the "best path(s)" can be computationally hard. In this thesis, we do not address this issue but the work in [9] helps solve this problem. Instead, we focus on evaluating paths that are requested by applications.

1.1.2 Monitoring the Internet

To collect dynamic path information that will be used to rate paths for applications, NPAS must monitor the Internet. To accurately rate paths while at the same time keep the measurement overhead at an acceptable level, NPAS not only needs to keep track of dynamically changed path condition, but also needs to decide monitoring points based on the application's requests.

Current measurement systems are often designed to measure a local network or a small scale network. Because the network being monitored is small, the system is able to monitor the entire network. NPAS, on the other hand, needs to provide information about end-to-end paths that span multiple ISPs. In other words, the measurement/monitoring system needs to scale to the size of the Internet.

In addition, traditional monitoring systems are often designed to be used by network administrators, not by applications. Network administrators usually have different interests in network metrics than applications. For example, in case of a link failure, network administrators may want to obtain network information that can help find the reason for the link failure, while applications don't care why the link failed, but rather only care about finding alternative paths that they can use to

send packets. Consequently, NPAS should look for dynamic end-to-end path metrics (e.g, bandwidth, latency) that applications are most interested in.

Unlike (local) monitoring systems deployed by ISPs, NPAS is designed to provide advice about end-to-end paths across the Internet. To achieve the desired scalability and timely collection/reporting of network measurements, NPAS leverages emerging programmable networks to dynamically adjust what it monitors based on current application patterns. While programmable networks enable NPAS to dynamically instrument the network with the measurement points needed by the current set of applications, NPAS must still reason out what path information needs to be collected and what nodes should be monitored while taking into account the network overhead, measurement overhead, and the ability/limitations of turning monitors on and off at network nodes. This reasoning ability enables NPAS to collect only path information that might be of interest to applications thereby reducing measurement overhead.

Since NPAS cannot monitor every path, NPAS will not have path information for some paths. In cases where NPAS does not have the complete path information for the requested path, NPAS should still evaluate the path based on partial path information (e.g., sub path information).

1.1.3 Collecting End User Feedback

In addition to collecting path measurements from ISPs, NPAS also collects feedback information from applications and uses feedback to evaluate network paths for future path queries. Feedback can provide path information about certain paths that could not be monitored otherwise. In addition, feedback can contain the application's choice of route. The application's choice of route can be used as the recommended route for other applications. For example, if a path is picked by some video applications, the same path could be recommended to another video application even if NPAS does not has path measurement information to evaluate the path. The application's choice

of route can also be used as an indication of whether applications are satisfied with NPAS's advice.

1.2 Contributions of the Thesis

The contributions of this thesis include:

- Envisioning routing in the future Internet: Routing services will exist in the future Internet. They will find connectivity between a source and destination, and will provide applications with a set of paths from which applications can choose. However, the routing services will not rank paths. The routes provided by routing services will contain a series of link (or node) identifiers which will help NPAS identify the monitor points.

- Designing a distributed NPAS architecture: Providing applications with path advice requires us to build a distributed NPAS system to collect and distribute path/link measurements, and to deliver path advice to applications. We introduce shared (global) NPAS servers and local NPAS servers to deliver up-to-date path information to applications and to share path/link information among ASes.

- Designing the NPAS APIs: In order for applications to use NPAS, NPAS should provide a set of APIs that rate paths, schedule traffic over multiple paths, and collect feedback. It is challenging to design the NPAS APIs so that the APIs can be used in different scenarios. We need to envision how applications will make use of NPAS in the future network. In addition, a well designed API can help NPAS collect path request information (which can help NPAS decide what path information should be collected) and feedback from applications.

- Monitoring the future Internet for applications in a scalable way: NPAS is designed to provide information about Internet-scale networks, not individual

ISPs or ASes. It is infeasible to monitor all paths. Therefore, NPAS needs to decide how many monitoring points are needed and where to place monitors. Since paths can span multiple ISPs (or ASes) and getting path information from ASes may be expensive, NPAS must keep the number of ASes (nodes) that it needs to interact with as small as possible. On the other hand, determining the monitoring frequency and how often the path/link measurements need to be distributed are also important to help NPAS scale.

- Testing and evaluating NPAS: We use real-world traffic traces to identify the number of measurement points needed by NPAS to efficiently and effectively monitor the Internet. Our analysis shows that current Internet traffic is highly concentrated on a small set of paths involving a relatively small number of links (popular links). As one might expect, the Internet traffic is largely destined for a handful of the major content providers (e.g., Google, Facebook, etc). In light of this fact, by strategically placing monitors in as few as 100 ASes to monitor 5000 shared links and a few local links per AS, NPAS is able to provide path/link information for about 70% to 90% of traffic. To further test the NPAS system, we used a simulated network with dynamic path conditions. The experimental results show that applications can greatly improve their performance (e.g, increase throughput) by taking NPAS-recommended paths.

1.3 Thesis Organization

We begin by describing existing network measurement techniques and traditional network measurement infrastructures for various types of networks in Chapter 2. Chapter 3 addresses the characteristics of the future network where NPAS will be deployed. Chapter 4 presents the architecture of NPAS, followed by the description of the NPAS service abstraction in Chapter 5. In Chapter 6, we study the existing traffic patterns and propose algorithms to identify monitoring points. Chapter 7 and

Chapter 8 discuss how to collect path information and rate paths for applications. In Chapter 9, we present a simulation model and the NPAS simulation results. Finally, we present our concluding thoughts in Chapter 10.

Chapter 2

Related Work

Historically there has been little need to provide network *applications* and *services* with information about the network's performance, because the network (or the network operator) makes the routing decisions, not the application. Consequently, traditional network monitoring systems [10] are largely designed to provide information for *network administrators*. There are a variety of existing network measurement infrastructure services [11, 12] that use active measurement tools (e.g., ping, traceroute, pathchar [13], pathload [14], pathrate [15], and iperf) and passive measurement tools (e.g., NetFlow collectors [16], nettime [17], and spand [18]) to collect network measurements. Although applications might be able to obtain path information from such tools, these tools are usually designed to help network administrators debug problems, optimize network performance, re-architect the network topology, or to generally keep the network running. As a result, many of the existing tools are designed to look for faults and signal alerts, or to look for anomalous behavior in the network that might indicate a potential failure or a security breach.

Unlike traditional monitoring systems, NPAS's primary goal is to assist network *applications* and *services* by gathering the information they need, which often differs from the information a network administrator needs. In particular, NPAS focuses on various *end-to-end* network metrics associated with *paths* (as opposed to traditional monitoring which is focused on the links and nodes managed by an ISP). Example

end-to-end path metrics include the available bandwidth of a path, the latency of a path, the loss rate of a path, or the variability and reliability of a path – information that must be collected across ISPs.

Tomography-based models, such as [19] and [20], can be used to calculate path/link information. However, tomography-based models fail to provide accurate and up-to-date path information (e.g., the current available bandwidth of a path). QoS routing [21, 22, 23, 24] has also been studied as a way for applications to find paths that meet applications' needs. However, QoS routing has its limitations. For example, QoS routing needs to reserve network resources. There is also some research [25, 26, 27] on finding paths for applications in overlay networks. However, these approaches do not address the issue of measuring performance of the paths in a scalable way.

2.1 Active and Passive Approaches

Network measurements that require injecting extra packets into the network are identified as *Active Network Measurement* approaches. Ping and traceroute are two examples of well-known active measurement tools. Performing active network measurement can interfere with normal network traffic. For example, when iperf is used to test TCP throughput between two end hosts, other TCP traffic sharing the same path may experience packet loss thereby reducing the sending rate. Therefore, reducing the measurement-related overhead is very important for systems that use active network measurement. Using fewer samples or using estimation and reasoning techniques to calculate the needed network characteristics can reduce the overhead, but may affect the measurement accuracy. Despite its limitations, the active measurement technique could be used (in a balanced way) by NPAS to collect network measurements between monitoring points, especially when these monitoring points are not directly connected to each other.

In contrast to active network measurements, passive network measurement does not inject extra packets into the network. Instead, passive network measurement tools often listen on network devices (e.g., routers), capture packets coming through the devices, and analyze the captured packets to obtain network information. For example, SNMP [28] data and netflow [29] data are usually collected passively. Passive measurement tools are good for measuring traffic statistics of a network device, such as the sending/receiving rate, types of traffic sent/received, etc.. The main advantage of passive network measurement is that it has very little affect on the network. However, it introduces overhead on the host system where the passive measurement tool is running, because various resources, such as CPU, memory and storage, are needed for capturing packets, processing packets, analyzing packets, and storing the final results. Since passive network measurement does not introduce measurement traffic to the network, making use of passive measurement techniques help NPAS learn about current network performance without imposing additional network overhead.

A challenge in passive measurement is the huge amounts of data that pass through certain nodes. Traditionally, collected data is stored in large databases and post processed to get traffic measurements. To do real time measurement, a measurement node should have enough processing power to generate measurement results on demand. Typically, users only request a few network measurements (e.g., statistics for heavy TCP flows). To efficiently get the requested measurements, filters can be used to extract the desired data. The distributed online measurement environment (DOME) proposed in [30] is an example of passive measurement system that is capable of handling measurement data in real time. Since DOME installs filters and collects statistics on measurement nodes according to users' queries, DOME is able to provide answers to queries in real time. The accuracy of DOME on certain queries depends on the amount of memory provided for those queries. In general, passive network measurement systems should focus on processing the observed traffic to get the desired

measurements while considering the resource consumption on the host system at the same time. Similarly, in NPAS, the processing overhead of the passively collected data can be reduced in two different ways. First, passive collection occurs only on nodes that carry measurements which are of interests to applications. Second, low processing frequencies are used for measurements that are infrequently requested.

2.2 Network Measurements

There are various path metrics that may be of interest to the application, such as latency, bandwidth, etc. There are many existing research works on how to perform network measurements and collect path information effectively.

2.2.1 Measuring Network Latency

Network distance can be measured in a variety of ways including latency, hop counts, geographic distance, etc. Ping and traceroute are two popular tools to measure network distance in terms of latency and hop counts respectively.

The latency between two hosts is an important network characteristic that, if reduced, can be used to provide better services to end users and to improve performance of applications. For example, a game server can match users with low latency paths, a content distribution network can locate the closest server for its customers by comparing the latency between servers and customers, and a P2P network can use latency information to organize peers.

Performing measurements between every pair of hosts to get distance information (in terms of latency) does not scale. A better approach is to try to predict the network distance. *IDMaps* [31] uses special *HOPS* servers to maintain a virtual topology map of regular hosts and special hosts called *Tracers*. If d_{AB} is the distance between host A and B, and T_A is the nearest tracer to host A, IDMaps estimates d_{AB} as follows:

$$d_{AB} = d_{AT_A} + d_{BT_B} + d_{T_AT_B}.$$

The accuracy of IDMaps can be improved when the number of Tracers is increased.

Global Network Positioning (GNP) [32] can also be used to estimate the Internet network distance. Every node in GNP has a coordinate, and the coordinate distance between two nodes is used as the estimated network distance. First, GNP picks N nodes as *Landmarks*. Second, when a host joins the system, the coordinate of the host is calculated in a way to minimize the error between the measured distance and the computed distance from the host to all the Landmarks. GNP takes a peer-to-peer approach, and each peer computes its own coordinate, which makes GNP easy to scale. One disadvantage of the peer-to-peer approach is that a host can lie about its coordinate. Another disadvantage is that landmark nodes could be overloaded if many hosts join GNP at about the same time.

The major difference between GNP and IDMaps is that GNP uses an absolute coordinate for each end host. The estimated distance between two hosts can be easily computed based on the coordinates. However, these approaches (e.g., IDMap and GNP) assume a stable network. The accuracy of prediction could be greatly affected by frequent network topology changes. GNP uses the estimation mechanism to deal with the scalability problem, and only provides an approximate distance between two end hosts. The estimated distance (in terms of latency) provided by GNP may not be useful for latency sensitive applications that require accurate latency values. The estimated latency may help NPAS give advice to applications on choosing paths whose latency do not change frequently. However, to give accurate path advice to applications on choosing paths whose conditions change frequently, NPAS needs to collect up-to-date and accurate latency information.

2.2.2 Active Bandwidth Measurement

Network bandwidth is another important metric for applications, especially for applications that have specific QoS bandwidth requirements. For example, a video

streaming server can dynamically adjust its streaming rate based on the measured bandwidth. Content can be distributed at a dynamic rate based on the available bandwidth in a *content distribution network (CDN)* [33]. A "backup" application may choose idle paths for replicating data. In addition, Internet users may want to check the bandwidth on the path they paid for.

Bandwidth can be measured hop-by-hop [34, 13, 35], or end-to-end [36, 37, 15, 38]. Hop-by-hop bandwidth measurement relies on the fact that the one-way delay (note that the one-way delay is often estimated using RTT to avoid installing special software in the router) changes as the packet size increases, and it requires routers to generate ICMP replies. Since intermediate routers may have different processing paths for replying to ICMP packets, hop-by-hop bandwidth measurement is often inaccurate. End-to-end bandwidth measurement requires cooperation between the sender and the receiver. There are two types of bandwidth to measure. One is *available bandwidth* (unused bandwidth) along a path, which is determined by the link with the minimum available bandwidth. The other is *capacity* along a path, which is determined by the link with the minimum capacity (raw bandwidth of the link).

Packet pair technology [15, 39, 40, 41] is one technique that is used in bandwidth measurement. However, the accuracy of packet pair technology can be affected by cross traffic. To address the cross traffic effect, Dovrolis et. al. [15] propose a method to estimate the capacity, and the method has been implemented in a tool called *pathrate*. Pathrate divides estimated capacity results into three categories: *Capacity Mode (CM)*, *Sub-capacity Dispersion Range (SCDR)*, and *Post-Narrow Capacity Modes (PNCMs)*. Estimation results in CM correctly represent the actual path capacity, while results in SCDR and PNCMs underestimate and overestimate the capacity respectively. The goal is to distinguish CM from PNCMs and SCDR. Their simulation results show that when the cross packet size varies uniformly, the dispersion

of the estimation results is less predictable. However, the CM is still distinct from PNCMs in the distribution of results.

Packet tailgating is another technique used in bandwidth measurement. Harfoush et. al. [38] introduce a method that makes use of packet tailgating to measure the capacity of an arbitrary subpath. However there exist certain conditions where subpath capacity cannot be measured. In addition, this approach [38] assumes that there is no cross traffic. Cross traffic can either lead to capacity underestimation or capacity overestimation.

The methods described in [15] and [38] focused on measuring the capacity of a path. In addition to the capacity of a path, an application may also need to know the available bandwidth of a path. *Spruce* [37] is a tool to measure the available bandwidth of a path. There are two main approaches used for available bandwidth measurement: the *probe gap mode (PGM)* and the *probe rate mode (PRM)*. Tools such as Pathload [14], Pathchirp [42], PTR [43], and TOPP [44] use PRM to measure the available bandwidth. Spruce uses PGM to measure the available bandwidth, and can effectively keep track of the available bandwidth due to its simplicity and lack of tunable parameters.

The bandwidth measurement techniques discussed above can be used by NPAS to measure available bandwidth between monitoring points, and to infer bottleneck links. NPAS can use these active bandwidth measurement techniques for paths/links on which NPAS can not measure bandwidth using passive methods.

2.2.3 Passive Bandwidth Measurement

Active measurements are usually used for measuring the available bandwidth of a path. However, under certain circumstances, passive measurement can measure the maximum throughput of a network path with acceptable accuracy. For example, Gerber et. al. [45] introduce a passive method to measure the maximum throughput

in a wireless network. Some of the traditional active bandwidth measurement tools do not consider the additional delay caused by a wireless network (e.g., packet loss due to wireless transfer, or the scheduling of wireless transmission). Compared to passive network measurements, traditional active measurement tools impose additional traffic on the network, are expensive to deploy, and do not consider that real users are at different vantage points in a 3G network. Passive max-throughput measurement in a wireless network has its own challenges. First, the TCP slow start is longer in wireless environments. Second, the flow may be rate limited by the content providers.

The passive max-throughput measurement in [45] is based on the analysis of TCP flow records collected passively, and can estimate the maximum throughput of wireless networks. However, the accuracy of estimation is affected by the way in which unwanted flows are filtered out. As noted earlier, bandwidth is an important metric NPAS would like to know to help applications select the appropriate routing paths. Making use of passive bandwidth measurement technologies can help NPAS get the path bandwidth information without introducing too much overhead on the network.

2.2.4 Application-specific Network Metrics

We know passive flow records can be used to estimate available bandwidth. However, an application may need to know other types of network information, which may require inspection of specific types of packets or flow records. Several application-specific monitoring systems have been used to collect application-specific network information. However, application-specific monitoring systems have several disadvantages. First, the implementation of multiple monitoring systems on routers is complicated, including the allocation of routers' resources to multiple systems. Second, applications change over time, as well as their network traffic patterns. To address these problems, the "minimalist" approach [46] can be used to monitor

network flow, which has several features:

- A few generic primitives. To reduce the router complexity, only a few primitives are imposed on routers.
- Separating collection from computation. To satisfy generic applications, computation needs to be performed separately from collection so that different analysis can be performed on the collected records.
- Network-wide management. Scheduling of monitoring on routers is necessary to provide network-wide management.

The "minimalist" approach [46] is a feasible rather than an optimal solution of adaptive passive monitoring system for applications. The experimental results show that the minimalist approach has comparable performance to application-specific approaches.

We learn from the "minimalist" approach that although NPAS is an application driven system, the design of NPAS needs to be generic to meet the requirements for various types of applications.

2.2.5 Packet-Level Traffic Measurement

The amount of network performance information that can be retrieved from flow records is limited. In some cases, packet-level traffic measurement is necessary. The benefits of packet-level monitoring include: determining the amount of over-provisioning in a network, studying traffic dynamics, detecting network anomalies, identifying network congestion and studying TCP behavior, and evaluating the network's capabilities to support new services (e.g., QOS) [47].

Fraleigh et. al. [47] propose a packet-level measurement architecture: *IPMON*. IPMON focuses on collecting packet-level traces from the Sprint IP backbone, and supports link speeds of up to OC-48 (2.5 Gbps). Each captured packet is marked

with sub-microsecond timestamps. The packet traces collected at multiple points are synchronized to within 5 μs. IPMON infrastructure includes three elements: a monitoring entity, a data repository, and a data analysis platform. Although IPMON will not scale to monitor every link in a tier 1 backbone, the current scale of IPMON can operate at a high speed and provide important data for understanding a network's dynamics. In the case that the detailed information on certain links/paths is required for some applications, the existing techniques (e.g., IPMON) provide NPAS with possible solutions to perform packet-level measurements.

2.3 Monitoring Various Types of Networks

There are measurement infrastructures that are designed for various types of networks. Analyzing the existing monitoring systems can help us design NPAS for the future Internet.

2.3.1 DNS based Measurement Infrastructure

The current Internet already has useful network-wide infrastructure that could be modified/used to support monitoring. The *Domain name system* (DNS) [48] is an example of such an infrastructure. ISPs may not want to deploy a new measurement system which has a different architecture from the current Internet due to the cost of using new devices and software. However, if a measurement system largely keeps the current Internet infrastructure as is, only introducing small modifications, then ISPs may accept and deploy the measurement system. DNS is a hierarchical naming system, providing mapping from human readable domain names to various data such as IP addresses. Because DNS servers are distributed throughout the entire Internet, they can be used to estimate delay between end hosts [49, 50].

In [49], the authors introduce a technology called *King* which relies on the DNS infrastructure to measure network distance. King assumes that end hosts

are close to their authoritative DNS servers that maintain their IP addresses. The latency between end hosts A and B is approximated using the latency between their authoritative DNS servers X and Y. King does not need the cooperation of A and B, and any King client can measure the latency between DNS servers X and Y. All that is needed is the ability to find out the authoritative DNS server for the end hosts.

Measurement systems such as King could have some potential drawbacks (e.g., King has problems with multiple DNS servers and DNS forwarders). However, the accuracy of King can be improved with small modifications to the existing infrastructure. *Turbo King (T-King)* [50] is an improved version of *King*. T-King assumes that it has control of some DNS servers, and it can modify the behaviors of DNS servers to perform an accurate latency measurement. T-King is able to deal with the problem of multiple DNS servers and detect if a DNS forwarder is used in the measurement process. Although T-King has advantages over King, certain DNS servers need to be modified for measurement purpose while King requires no modification of the DNS.

Note that the above DNS-based measurement systems cannot measure certain path information. For example, measuring the bandwidth between DNS servers provides little to no information about the bandwidth available between end systems. Moreover, even the latency measurement between DNS servers and end hosts requires active probes (ICMP messages) done by end-systems which could result in significant measurement overhead.

2.3.2 Overlay Networks

An *overlay* network is a computer network built on top of another network. The links in the overlay network are considered to be virtual or logical, consisting of one or more physical links in the underlying network.

Because overlays are built above the IP layer, overlays can develop and use any addressing, routing, and forwarding algorithms they wants. For example, *distributed hash tables (DHT)* (e.g, Chord [51] and Tapestry [52]) can be used for routing in an overlay network. In addition to the DHT protocol, many other protocols, such as JXTA [53], XMPP [54], Freenet [55] Tor [56], and Gnutella [57], can be used in the overlay network for communication between peers. However, the overlay network does not have control on how the underlying network routes packets between overlay nodes.

There is some research (e.g., BARON [58]) that studies the path metrics for different paths in overlay network to select paths between peers. BARON [58] assumes a relatively small number of overlay routers and thus takes a brute-force approach to monitoring, intrusively monitoring the virtual links between every pair of overlay routers which does not scale and imposes a significant monitoring load on the network. NPAS, on the other hand, focuses on determining a small set of links/nodes that need to be monitored based on the current needs of applications.

2.3.3 Virtual Networks

A virtual network is a computer network that consists of virtual resources (e.g., virtual links, virtual machines, virtual routers). There are two types of virtual networks: *protocol-based virtual networks* and *virtual-device-based virtual networks*. Protocol-based virtual networks include *Virtual Local Area Networks (VLANs)* [59], *Virtual Private Networks (VPNs)* [60], *Virtual Private LAN Service (VPLSs)* [61], etc. Virtual-device-based virtual networks are networks connecting virtual devices (machines), such as Emulab [62], Planetlab [63], VINI [64], and GENI [3].

The monitoring infrastructure for a virtual network should not only provide interfaces for the network administrator, but also for network-aware applications so that the applications can benefit from the monitoring infrastructure [65]. Ciuffoletti [65]

proposes a solution to make a network monitoring system on a virtual network configurable by applications. Obviously, one does not want an application to fully control the network monitoring system for security concerns. Fortunately, network monitoring tools can be designed as plug-ins to virtual hosts.

Monitoring systems on virtual networks are often constrained or influenced by the virtual environment. For example, physical resources are often shared by multiple experiments in virtual networks like Planetlab, and the monitoring system must also share (i.e., run on top of) the physical network resources. Resource scheduling needed by shared environments often leads to difficulty in gathering fine-grained timing information. Timing is important when calculating, for example, the difference between the time the request packet was sent and the time the reply packet was received, needed to compute the RTT between two nodes. When measuring the RTT in the Planetlab environment, there can be a delay between the arrival of the reply packet and the processing of the reply packet, leading to an inaccurate RTT estimation because the resource that the measurement process needs is unavailable at the time the packet arrives.

The *multi-user active measurement system (MAD)* [66] can be used to solve the timing problem for performing measurement in a shared environment. The idea behind MAD is to have certain measurement code running at the highest priority level so as to gain accurate timing. As a measurement system running on virtual networks, MAD also takes features of the virtual network into account thereby supporting multiple users, imposing low impact on shared nodes, and having flexibility for different measurements.

The Paths requested by applications can be a virtual path or a path that goes through a virtual network. Consequently, understanding virtual network technology can help NPAS measure and evaluate these paths.

2.4 Measuring the End-to-End Path

Tomography-based models [19, 20, 67, 68, 69, 70, 71, 72] can be used to calculate end-to-end path information. However, tomography-based models often assume a static network matrix and calculate link information based on estimation rather than directly measuring the link. In a network where the path conditions change frequently over time, applications need a monitoring system that can provide up-to-date path information to choose the appropriate paths. In addition, the number of nodes that need to be monitored by network tomography models [73, 74] is still far higher than the number of nodes NPAS expects to monitor (e.g., 100 nodes globally).

There is also some research on finding routing paths for applications in overlay networks [25, 26, 27]. These approaches described in [25, 26, 27] focus on finding a small set of paths for applications with the help of topology information rather than paying attention to the evaluation of the paths being found. Nakao et. al. [25] propose a routing underlay. When making application-specific routing decision, overlay networks query the routing underlay. The routing underlay, in turn, asks the underlying Internet for topology information. Although in [25], the authors have realized the need of making informed application-specific routing decision, they do not address the issue of measuring possible paths and helping applications choose the appropriate paths. The *one-hop source routing* infrastructure [27] describes a method to find k alternative paths between two nodes, but it probes all k paths that have been found before a final path can be chosen. The one-hop source routing approach leads to the scalability problem if all applications need to make such probes before they send out each traffic flow. Fei et. al. [26] propose an approach for selecting a good alternative path from a large number of available candidates. However, this approach [26] finds the alternative path based on how the path is divergent from the default path, and tries to reduce the likelihood that the alternative path quickly merges back into the default path by choosing relay nodes that are "far" from the

default path. The approach in [26] does not measure the alternative paths and use the end-to-end measurements to select a path that meet the application's requirements. On the other hand, NPAS assumes that there already exist a small set of paths, and designs a scalable approach to measure these paths.

2.5 QOS Routing

Quality of service (QoS) is important for many applications. The *Resource Reservation Protocol (RSVP)* [75] is a transport layer protocol which can be used to provide QoS for applications. RSVP is not a routing protocol. Instead, RSVP is designed to interact with the current or future routing protocol to reserve resources across a network. Because the resources in a path are reserved, an application with specific QoS requirements can be satisfied when the application uses the path. In order for RSVP to work, the routers in the path, on which the RSVP tries to reserve resources, need to understand the RSVP protocol and be able to reserve the required resources.

RSVP is a receiver driven protocol. The receiver initiates a resource reservation request along the path to the sender for a flow. The request includes the *Flowspec* and the *Filterspec*. The Flowspec contains the QoS requirements of a flow, and routers can schedule packets based on the Flowspec. The Filterspec tells routers how to identify packets that should be affected by the Flowspec. RSVP requires the router to maintain soft state of resource reservations. To keep resources reserved in a path, the soft state needs to be periodically refreshed by a path message (a path message is sent by the sender to set up a reversed path) or a reservation request message.

There are several disadvantages of using RSVP to provide QoS for applications, including:

- Routers have to implement RSVP to reserve resources, which leads to a complicated router design.

- Routers have to maintain state information for RSVP, which causes scalability problem.

Yang et. al. [76] provide a solution to End-to-End QoS guaranteed routing. The approach in [76] assigns a certain amount of bandwidth to each edge router. The edge router allocates assigned bandwidth to flows that are originated from that edge router. By having each edge router remember the usage of its assigned bandwidth, an edge router can make decisions to accept/reject flows instantaneously without hop-by-hop signaling. However, the edge routers need to maintain the state of assigned bandwidth, and the assigned bandwidth at edge routers needs to be updated when the bandwidth demand is changed.

In short, existing QoS routing approaches (e.g., RSVP) consider path characteristics (e.g., available bandwidth) when selecting paths. Routes are selected based on what can be reserved/guaranteed and thus these approaches focus on the problem of simultaneously *finding* and *reserving* a path with sufficient resources. NPAS, on the other hand, is designed to give information/advice about the behavior of all paths specified in the request. Moreover, because QoS approaches require control of the routers along a path (i.e., to make reservations), they are often limited to a particular domain and cannot be used end-to-end.

Chapter 3

Future Network Environments

While the current Internet Protocol (IP) supports loose source routing, it is not widely used and in many cases is not supported by ISPs. However, current overlays and many proposed future network architectures support source routing – or at least path selection. NPAS is designed for these types of networks.

To empower applications with the ability to select paths across the Internet and enable NPAS to collect measurement data from the network, we must first explain the types of future networks that we envision NPAS being used with. In particular, we need to explain how monitoring will be supported in future networks, how paths could be represented by the routing service, and the role of the routing service.

3.1 Support for Monitoring in the Future

A path can span multiple ISPs. Although ISPs may want to hide path performance information within their domains, ISPs may provide inter-AS link performance information such as latency (including the queuing delay) and available bandwidth on the inter-AS links – links that connect two ASes. Providing such information does not reveal the intra-domain routing information, but can help attract traffic to an AS. In the model described in *ChoiceNet* [77], ISPs are paid based on the amount of traffic that goes through their ASes. Providing AS-level link information to NPAS can result in NPAS recommending paths that traverse that link. Consequently, the

AS can make money by advertising its inter-AS links. Alternatively, NPAS can pay ISPs for certain measurement data.

To collect end-to-end path information, NPAS needs (direct or indirect) access to monitoring points, controlled individual ISPs (we do not assume NPAS has control of monitoring points). Consequently, NPAS depends on individual ISPs to provide accurate path/link information. If an AS intentionally provides inaccurate path/link information, NPAS could find out whether the path information is correct with the help of end-to-end measurements (e.g, feedback from applications) and measurements between trusted ASes. For simplicity, in this thesis, we assume that ASes who provide path information to NPAS give out accurate information. Since it could be expensive for NPAS to set up service level agreements (SLAs) with ISPs, NPAS limits the number of ASes (monitoring points) from which NPAS needs to collect path information.

We expect NPAS will collect network information that most applications are concerned about, such as latency and bandwidth, from the monitoring points. Various measurement tools can be used in future networks to get path information. For example, the commonly used program ping can be used to gather latency information between network nodes. ISPs can use whatever measurement tools they want on the monitoring points. In addition, ISPs may allow customers to deploy measurement tools. For example, ISPs may support new service infrastructure, such as *infrastructure as a service (IaaS)* [78] (e.g., Amazon EC2), and *platform as a service (PaaS)* [79] (e.g, Google App Engine), which allow users to run their own measurement tools or support APIs that can be used to develop measurement tools. NPAS can also make use of these services to collect path information that is of interest to applications.

To monitor the Internet in a scalable way, we assume that NPAS has the ability to enable/disable monitoring at various monitoring points in the network (but not

necessarily at all monitoring points). We expect ISPs will provide APIs that can be used by NPAS to inform ISPs that monitoring on specific links is no longer needed. Emerging network technologies, such as programmable networks, allow dynamical control of the network nodes. As an example, consider the GENI network [3], in which nodes can be customized to dynamically load and run node-specified programs. In the future network, similar technologies may be used to enable NPAS to dynamically turn on/off monitor. Alternatively, we may have something like the current Internet with SNMP ability to interact with and control network monitoring points.

3.2 Network Topology and Paths

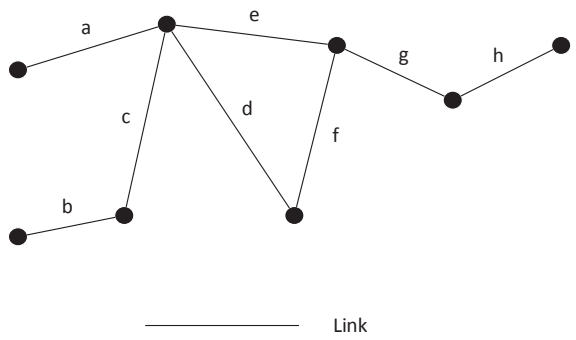

Figure 3.1: A Network Topology

In order for the routing service to return *paths* that can be understood and used by applications, we need to define a way to represent paths. Towards that goal, we assume that the network topology can be represented by an undirected graph of nodes and links. A network node relays packets from one link to another, while a link transits a packet from one node to another. Moreover, we assume that each link is identified by a unique identifier[1]. An example network topology with identifiable links is shown in Figure 3.1. In today's Internet, one could uniquely identify a link using a pair of IP addresses assigned to the endpoints of the link. In future network architectures,

[1]Note that one could (alternatively) identify nodes instead of links.

we may see a variety of different addressing schemes such as the addressing approach used in the Postmodern Internet Architecture [7] which assigns each link a unique channel ID. Given a topology of nodes and links, we assume that a network *path* is described as a sequence of identifiers that represent the links along the path[2]. For example, a path P, which goes through links a, e, g, and h, can be represented as $a - e - g - h$.

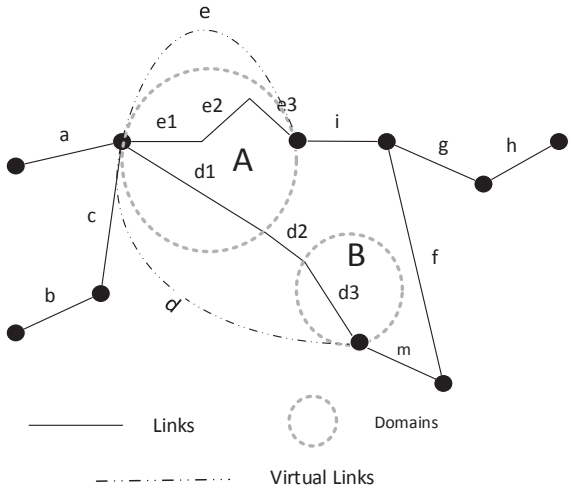

Figure 3.2: A Network Topology With Virtual Links

Because ISPs may not want to reveal information about their internal paths, we assume that some links may be *virtual links*: imaginary/logical links that have some (unknown) mapping to physical links. Virtual links can be used to represent intra-domain paths without specifying how packets are routed from one node to another. For example, link e and link d in Figure 3.2 are virtual links that consist of other links. As shown in Figure 3.2, virtual link e consists of link $e1$, link $e2$, and link $e3$. Link e is used to represent a route across domain A without revealing the underline physical links. Similarly, virtual link d consists of several links: $d1$, $d2$, and $d3$. Instead of representing a route across a single domain, link d is used to represent a route across

[2]Alternatively, one could use a series of nodes to represent a path

multiple domains: domain A and domain B. Virtual links can be recursively defined. In other words, link $d1$, $d2$, and $d3$ can also be virtual links.

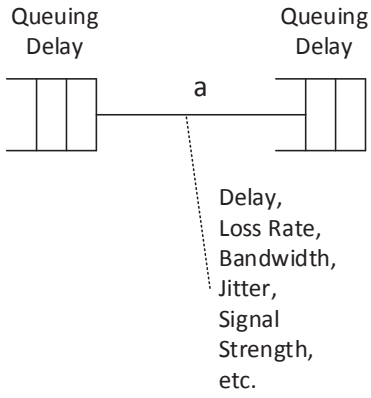

Figure 3.3: Link Characteristics

As shown in Figure 3.3, a (virtual) link has many metrics, and applications are interested in certain link metrics such as latency (including queuing delay), bandwidth, jitter, loss rate, and signal strength (for wireless links). Since a path consists of one or more links, NPAS needs to collect link measurements in order to evaluate the path for applications.

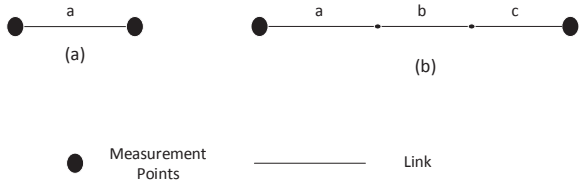

Figure 3.4: Network Measurement Points

To collect link measurements, NPAS needs to place monitoring points on the endpoints of links, as shown in Figure 3.4a. We assume that given a link name, the two endpoint nodes that are connected by the link can be identified. Therefore, given a path that consists of links, it is possible to identify nodes to place monitoring points on. Although it might not be allowed to place monitoring points on certain nodes, placing monitoring points on the neighboring nodes can help NAPS measure

the path across the links that do not enable monitoring points. For example, as shown in Figure 3.4b, although link b does not allow monitoring points, monitoring points on link a and link c can measure the path information from link a to link c.

Because paths can consist of both physical and virtual links, we assume that NPAS must be able to return information about paths that contain virtual links (even though NPAS may not know the way in which these paths are mapped onto physical links). For example, in Figure 3.2, *e-i-g-h* is a path, and link e is a virtual link on the path. NPAS may still be able to provide some information about the end-to-end performance of path *e-i-g-h* without knowing how link e is mapped to physical links.

Two links are said to be adjacent if they connect to the same node. A *contiguous path* is a path where every pair of links in the path are adjacent. In Figure 3.2, path *e-i-g-h* and path *a-e1-e2-e3-i-g-h* are contiguous paths. However, path *a-g-h* is not a contiguous path since link a and link g are not connected to the same node. A non-contiguous path is called a *partial path* because some links in the path are not adjacent. Partial paths are useful for representing paths with missing links. For example, in the extreme case, one might specify a path that only gives the outgoing link from the source and the incoming link at the receiver with all other links in between missing. While NPAS may not be able to do much with such paths, it should still be able to provide whatever information it has for the partial paths.

3.2.1 The NPAS's View of Topology

NPAS is not a routing system. NPAS does not know the topology or have any way of requesting topology information like a routing system does. However, NPAS does have access to certain ASes in the topology to turn on monitoring, but this is different than being able to gather routing information from these ASes. NPAS will gradually learn the topology by piecing together path requests. (This assumes

that NPAS receives path requests that have been created by the routing services and that applications have not modified/corrupted them. Routing services might need to "sign" their paths). In addition, NPAS does not participate in any routing protocols. NPAS is not designed for any particular metric (e.g., latency, bandwidth, etc.), but instead can request measurement information from certain ASes about these metrics and it knows how to combine them using either additive or max/min composition of values. New types of metrics (e.g., jitter, loss rate, etc.) could be added without changing the way the system works. So there is no overhead needed to collect routing information, and there is no convergence needed to stabilize on routes.

Routes that are provided by ISPs might have different granularity about individual links of a route. For example, ISPs may use one virtual link to represent a path across a domain or across multiple domains. Alternatively, ISPs can provide more detailed information about an intra-domain route by providing several inter-connected intra-domain (virtual) links in the route. NPAS will simply build a network topology at the granularity ISPs provide.

In this thesis, we assume applications can choose the AS-level path. The AS-level routing policies are enforced by the routing system, not NPAS. NPAS is designed to select from the set of paths allowed by the routing system. In that sense, NPAS does not violate AS-level routing policies when it selects an AS-level path. NPAS's job is to collect the inter-AS link information and eventually calculate the end-to-end AS-level path information. Individual ASes can have their own methods of controlling intra-AS routes, and NPAS does not help applications pick intra-AS routes.

The routes returned by the future routing service can be represented as a list of AS-level links that connect different domains. Typically, the endpoints of the inter-AS links are ingress points and egress points of connecting domains. In the current Internet, each AS can be considered as a domain in the network topology. As described in Figure 3.5, a route from the sender to the receiver goes through $AS1$,

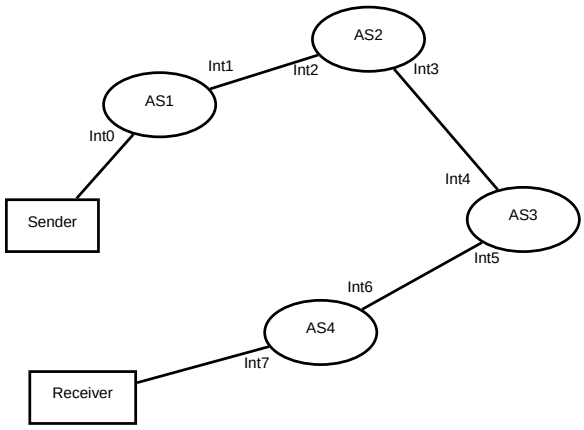

Figure 3.5: A Simple Network Topology With Monitoring Points

$AS2$, $AS3$, and $AS4$. $Int0$ to $Int7$ are the endpoints of the incoming and outgoing edge links of the ASes. The routing service may only be able to tell applications that the route goes through 3 AS level links: $AS1-AS2$, $AS2-AS3$, and $AS3-AS4$. By placing monitoring points on the endpoints, NPAS can obtain the link information about 3 AS level links: $AS1-AS2$, $AS2-AS3$, and $AS3-AS4$. Although NPAS does not focus on the intra-AS routes, NPAS can still estimate the condition of intra-AS routes if NPAS has the ability to perform measurements between monitoring points. For example, by performing measurement between monitoring points Int0 and Int1, NPAS can estimate the condition of the route across domain AS1. Because ISPs can dynamically change intra-AS route in case of congestion on intra-AS links, we focus on monitoring the inter-AS links in the rest of this paper.

3.3 The Role of Routing in the Future

As noted earlier, the future Internet is likely to allow applications to choose routing paths. To provide applications with the ability to pick end-to-end paths, the future routing service will need to find connectivity between nodes. In other words, an application will rely on the routing service to find a set of possible paths between two nodes. It is the routing service's responsibility to verify that the returned paths

exist. We do not assume that the routing service is able to rate paths based on their real time QoS, because there are a huge number of paths network-wide, and it is infeasible for the routing service to keep track of the fast changing path information of all paths. Therefore, NPAS is designed as a supplementary service to the routing service. Applications that need further advice on path selection can make use of NPAS.

This implies that the routing service will not need to find the "best route". The concept of "best" will be something that NPAS is responsible for. While this simplifies future routing services (i.e., future routing service does not need to pick the final path for applications), the new requirements of finding and returning multiple paths will add some complexity.

3.3.1 Finding a Set of Possible Paths

Future routing services are not expected to select a single "best" path between a source and destination as is currently done by protocols such as BGP [1], but rather future routing services will be designed to return a *set of paths* that are assumed to be up and capable of providing connectivity between a source and destination. The simplest routing service may only be concerned with connectivity, with no ability to rank paths at all. However, even these simple services will need to limit the number of paths they find/return to applications because the number of paths can quickly grow to be very large. Future routing services are unlikely to maintain fast changing QoS information that would help applications determine routes. Nor will routing services pick the best path (or paths). Some important network metrics that greatly affect the performance of applications, such as available bandwidth and latency, change dynamically. Because of measurement overhead and processing overhead, it is infeasible for the routing service to monitor all the paths in a network to collect dynamical network information that is required to select paths for an arbitrary

application.

If no ranking information is available, the routing service would select a subset of the paths based on its own internal algorithm (e.g., a depth-first search of the topology, a route selection based on randomly chosen intermediaries as described in [27], or a random selection from the list of all known paths). However, in general, we assume that routing services will make use of static link characteristics (e.g., static link capacity) and the number of links in a path to rank paths and then return the top paths. For example, Ascigil et. al, [9] describe an approach that computes paths using static QoS information, and has some ability to integrate fast changing QoS measurements in path computation (if they were somehow available). Another example method of finding a small number of paths between a source and destination is to return paths based on the number of links in the path. Routing services can sort the paths firstly by the number of links and secondly based on the static capacity of the links. These sorted paths can be used as the candidate paths when routing services provide applications with the possible paths. However, one can imagine routing services that attempt to distribute load by returning continually changing subsets of paths that ensure applications are not always selecting from the same set of paths.

3.3.2 Specifying Path Queries

Applications might interact with the future routing service in the following way: applications tell the routing service how many paths they would like to know about, and get back a certain number of paths. However, the routing service does not find the "best" paths for applications.

We assume that paths returned by the routing service are contiguous paths. Contiguous paths enable NPAS to uniquely identify a path. Note that internal

network details can still be hidden by the routing service by returning contiguous paths that contain virtual links.

An example API for a future routing service is described as follows:

- Input: the number of paths that are requested by applications
- Output: a list of (AS-level) paths that are sorted firstly by the number of virtual links in the paths and secondly by the static link capacity.

In our envisioned networks, applications can ask NPAS to evaluate the contiguous paths that they obtain from the routing service.

Chapter 4

A Network Path Advising Service

After a routing service has identified a set of viable paths between a source and destination, it becomes the task of the *Network Path Advising Service (NPAS)* to rank or recommend which path (or paths) from the set are the best for the application. Defining such a ranking requires gathering information from the application in terms of its requirements and desires and requires information from the network in terms of its current performance. In the following, we describe example ways in which NPAS might be used, present a brief description of NPAS's features, and then provide an overview of the NPAS system.

4.1 Using NPAS

We envision applications issuing NPAS API calls to rank or rate the set of paths found by the routing service. An application sends the set of paths returned by the routing service, together with its own path requirements (e.g, the minimum amount of bandwidth needed), to NPAS asking it to rank the paths from best to worst. NPAS then comes up with a ranking/rating based on the application's specific requirements. Based on the path ratings, an application can then decide which path(s) it wants to use.

While the path ranking features of NPAS are largely focused on helping senders select paths, NPAS also offers services to receivers. In particular, a receiver can

inform NPAS about the paths it prefers to receive traffic from. For example, the receiver may have several major incoming paths, and wants the incoming traffic to be distributed across the available paths according to some rules (e.g., the receiver may limit the amount of traffic it wants to receive on each path). The receiver sends its requirements to NPAS. Subsequently, when NPAS rates routing paths for the sender, NPAS can also take the requirements of the receiver into consideration.

An application does not need to use NPAS on a per-packet basis. Instead, an application will use NPAS on per-flow basis. In addition, an application can also cache the advice provided by NPAS and continue use the same advice for new flows as long as the application does not experience poor performance of using the previous recommended path.

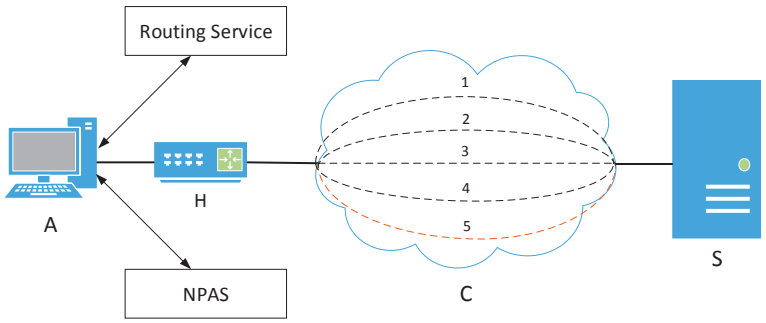

Figure 4.1: NPAS Use Cases

To understand how NPAS works, let's consider the example illustrated in Figure 4.1. Suppose S is a video streaming server, A is a customer of S, and H is A's gateway router. A wants to watch video that requires $1Mbps$ bandwidth. Suppose A has the ability to choose a path in the cloud C, and A wants to pick a single path to communicate with S. A first queries the routing service for possible paths across C. The routing service returns path 1 to path 4 to A. A then uses NPAS to evaluate paths 1 to 4, and A also specifies the available bandwidth requirement (e.g., $> 1Mbps$) and latency constraints. The path evaluation results from NPAS

may be presented as a ranking. Suppose that path 2 has the highest ranking. A can simply choose path 2. Alternatively, A can request detailed path information for path 1 to 4, and make its own decision to select a path. After a path is chosen, A can observe the path performance characteristics by using the path. A can then send feedback about its experience of using the path to NPAS.

Although path 5 is not included in the request, NPAS may have learned about path 5's existence and performance from other applications or previous queries. NPAS may have information about path 5, and believe that path 5 may also fit A's needs. Suppose that NPAS, in addition to rating paths 1 to 4, also recommends path 5. A may decide to use the path 5 or stick to the set of paths it asked about – i.e. to stick with path 2.

4.2 NPAS Features

While NPAS does not assume that every router is programmable or configurable, for those routers that are controllable by NPAS (possibly with the help of ISPs), NPAS should be able to dynamically turn on and off monitoring. Because NPAS is driven by requests from applications, NPAS knows what network performance information is needed to support applications, and can make intelligent decisions about what needs to be monitored and how frequently data should be gathered.

By virtue of being asked, NPAS learns that a sender is interested in the requested paths. Because monitoring network paths consumes network resources, it is not scalable for NPAS to monitor all the paths in a network. To deal with the scalability problem, NAPS does not promise or guarantee that it will evaluate all paths. However, the expectation is that the amount of information that needs to be collected frequently on small timescales is relatively small compared to the amount of information that does not require frequent updating. For example, information on paths that are not used or infrequently used by applications does not need to be collected frequently. In

other situations, network performance information may be requested by applications, but does not change quickly. For example, the capacity of a link will not change and thus only needs to be collected once. The goal is for NPAS to focus monitoring on the areas of the network that will help it answer the largest number of questions. In general, by focusing on popular paths and only monitoring the important changes, NPAS can keep the monitoring load to an acceptable level.

Although NPAS is not specifically designed for network administrators, network administrators can also benefit from NPAS. For example, the updated path statistics that are provided by NPAS may help network administrators determine whether there is something wrong with the path. However, network administrators may still want to perform additional measurements on certain nodes/links to identify the problem.

NPAS can use several methods to collect path information. In addition to monitoring the network directly, feedback from applications using the network can also be an important source of path information. The feedback may include specific performance metrics such as the sending rate an application actually reached, or just the application's final decision about the routing path (i.e., which path it ended up choosing after asking NPAS for advice). Feedback not only helps NPAS evaluate paths for which it has no monitoring data, but also helps NPAS reduce the number of measurements that need to be collected when feedback from applications has already provided the information needed to make accurate ratings of paths.

4.3 The NPAS System Architecture

Figure 4.2 illustrates the architecture of the NPAS system. At the heart of the system is the core NPAS service that is operated by the core NPAS service provider. Although companies may provide core NPAS services targeted at different types of applications (e.g., bandwidth sensitive applications, security sensitive applications, etc.), we expect that there are only a small number of (e.g., less than 10) core NPAS

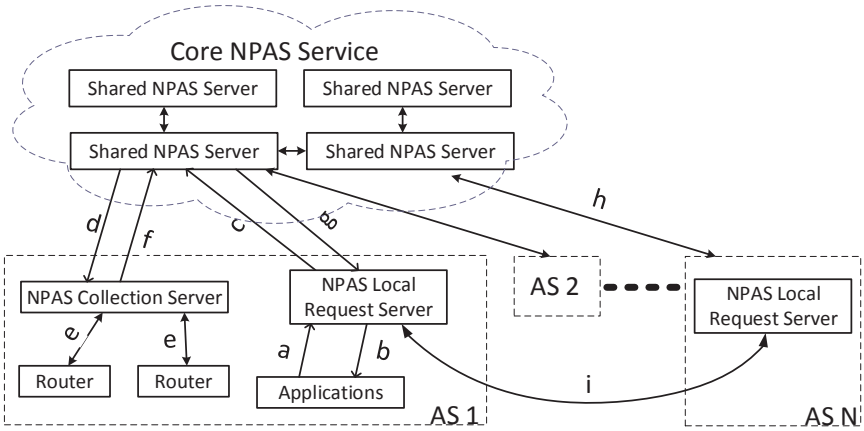

Figure 4.2: The NPAS System Architecture

service providers offered globally (i.e., NPAS service providers that are targeted at the global Internet.) [1]. The core NPAS service is responsible for gathering measurement data from routers through the *NPAS collection server* (indicated in lines e and f) and distributing measurement data to the *NPAS Local Request Server* (line g). The core NPAS service consists of a collection of *Shared NPAS Servers*. There could be dozens of shared NPAS servers that are distributed across the Internet for a core NPAS service. Each shared NPAS server is responsible for collecting data from a certain number of NPAS collection servers and distributing data to certain NPAS local request servers. The shared NPAS servers communicate with each other to aggregate measurement data collected from NPAS collection servers. A shared NPAS server is a server cluster that consists of multiple physical servers. Some of the physical servers can be used as the backup servers to improve the reliability of the NPAS system.

To start using the NPAS service, applications (clients) issue NPAS requests through a *NPAS Local Request Server* (line a). The NPAS local request server uses path information distributed by the shared NPAS server and some locally available path information to offer path advice for applications (line b). The NPAS local request

[1]The remainder of the thesis will describe NPAS in the context of a single core for the purpose of clarity.

server can be deployed by any AS who wants to provide path advice to applications. Although we do not assume NPAS local request servers will share their local path information, it is still possible for a NPAS local request server to get some local path information from other NPAS local request servers (line i) if the trust relationship can be established. The NPAS local request server periodically forwards applications' requests in an aggregated manner (e.g., once per 30 minutes) to the shared NPAS server (line c) in order to help the shared NPAS server find out what data are being requested by applications. Alternatively, the local request server can choose to send aggregated requests immediately without waiting until the next update in case that there is a significant reduction (e.g., 10% reduction) in the percentage of traffic that NPAS can offer advice for. The shared NPAS server, in turn, after detecting change of the requested paths, determines the new set of links/paths to be monitored. The NPAS local request server also helps hide the identity of the application by acting as a proxy between the shared NPAS Server and the application. The aggregated requests contain information about paths that are requested by applications, what metrics are of interest to applications, and how many times these metrics are requested on the path, but does not contain information about the exact time of individual requests and the specific requirements of a request such as the amount of bandwidth being requested or how long will the application use the path. As a result, while aggregated request information helps the core NPAS service find what paths/links need to be monitored, it does not reveal detailed request information of the source AS. Furthermore, the NPAS local request Server can choose to remove requests that are considered to contain sensitive data from the aggregated requests to protect privacy, or choose a trusted core NPAS service.

On the monitoring side, the shared NPAS server relies on NPAS collection servers to turn on monitoring (line d and line e), gather measurement data (line e), process measurement data, and then forward processed data to the shared NPAS Servers

(line f). NPAS collection servers are used to control data collection on specific routers/nodes in the network. For example, the NPAS collection server talks to individual nodes/routers to set up measurements, collect and aggregate measurement results, and then sends the (aggregated) measurement results back to the shared NPAS Server. It is expected that ASes would each have (or run) their own NPAS collection server that would determine what information it sends to the shared NPAS servers, and also determine what control over the measurement process it allows the shared NPAS servers to have (note that the shared NPAS server may only need to communicate with a small number of NPAS collection servers in order to provide path advice for the majority of requests). NPAS collection servers also help reduce the amount of traffic that is sent to the shared NPAS server by sending processed data (e.g., computing the average value of measurements over a time period) instead of raw data.

After getting the measurement data from various numbers of NPAS collection servers, the shared NPAS server needs to distribute measurement data to NPAS local request servers (line g). The NPAS local request server is also responsible for collecting the application's feedback. Because applications often communicate with the same destination over and over (i.e., use the same path over and over), the NPAS local request servers can cache path information and thereby enable fast response for requests for the same path. The timeout for information stored in the cache is determined based on the frequency at which the measurement is performed.

In order for the shared NPAS server to communicate with the NPAS collection server and the NPAS local request server, they need to authenticate with each other. We expect that the public-key cryptography based approach can be used to establish symmetric session keys that can be used for communications between the shared NPAS server and the NPAS collection server (or the NPAS local request server).

Since there are more than 40000 ASes currently in the Internet, the shared NPAS

server may need to distribute measurement data to more than 40000 NPAS local request servers. However, the shared NPAS server does not need to distribute measurement data at the same rate to all local request servers. The measurement data distribution rate and the potential network overhead involved in the NPAS system will be discussed in Section 6.5.

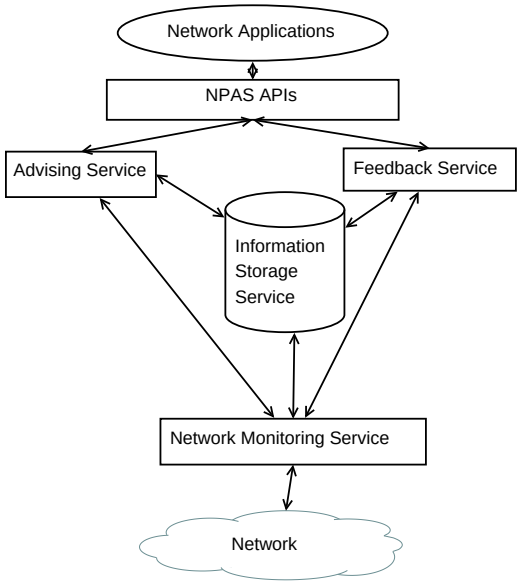

Figure 4.3: NPAS Components

A NPAS system may contain different service components to support network monitoring, path information collection/distribution, and path advice generation. Figure 4.3 provides an example view of the NPAS service components. There are four NPAS service components in a NPAS system: (1) *the information storage service*, (2) *the feedback service*, (3) *the advising service*, and (4) *the network monitoring service*. The shared NPAS server, the NPAS local request server, and the NPAS collection server work together to support these NPAS service components. The information storage service acts as a shared storage between all NPAS service components. The feedback service tries to collect feedback needed by NPAS from applications. The advising service evaluates routing paths based on specific requirements and provides advice on choosing and using paths to applications. The network monitoring

service reasons out what paths and nodes should be monitored, schedules actual measurements, and collects path statistics from the nodes being monitored.

Chapter 5

The NPAS Service Abstraction

A key aspect of NPAS is the *service abstraction* it presents to applications that use the service. The abstraction is based on the premise that applications will need help selecting and using paths, but will also contribute information back to the system about end-to-end performance. In particular, applications may interact with NPAS in three different ways. Consequently, NPAS supports three different Application Programming Interfaces (APIs) that programmers can use to interact with NPAS.

In the first case, applications will need help evaluating a set of paths returned by the routing service based on the requirements of the application. In this case, applications will provide NPAS with the application's communication requirements and a set of paths. NPAS will then return a ranking of the paths. We call this the NPAS *Rating API*.

A second way in which applications will interact with NPAS will be to request information about how best to use a path or set of paths. For example, an application may want to know how to schedule or multiplex packets across multiple paths to achieve the best performance. We call this the NPAS *Scheduling API*.

Finally, applications will report their experiences using paths back to the NPAS system to help it make more informed decisions in the future. Applications are in the best position to evaluate the end-to-end performance of a path and can provide this information in the form of feedback to the NPAS service. We call this the NPAS

Feedback API.

5.1 The Rating API

The rating API is designed to help applications identify the best paths. Both senders and receivers can make use of the rating API. The rating API can provide applications with a ranking and a rating. A ranking is an ordered list of paths, ranked from best to worst for some metrics. A rating includes information about a path and may include information about several metrics. The rating API may come back with rating information together with the ordered list of paths or NPAS may choose to hide the rating information. To rank or rate paths that are not completely monitored by NPAS, partial path information and feedback from applications are used to calculate ranking/rating results. In addition, the rating API may recommend paths that are not requested by applications. Over time NPAS may have learned about paths that could potentially satisfy the application's requirements but are not requested by applications. NPAS can also rate and recommend those paths to applications.

To appropriately rank (and rate) paths for an application, NPAS needs to know the set of paths from which an application can choose along with the application's Quality of Service (QoS) requirements. If PS_1 stands for a set of paths which an application can pick from (PS_1 can be represented as a list of paths), the input and output of the first form of rating API (the rating API for senders) is described below:

- Input: PS_1, list of QoS requirements
- Output: PS_2, RS_2

The list of QoS requirements can be formatted as a list of type and value pairs. The type can be bandwidth, latency, or other metrics that NPAS supports. The type can also be special information that applications would like to inform NPAS about, such as "time" which tells NPAS how long the paths will be used and "algorithm"

which tells NPAS which algorithm should be used to rank the paths (e.g., rank the paths based on a weighted combination of QoS metrics). The value indicates the requirement for a specific type. For example, if type is bandwidth, the value can be set to 1Mbps to indicate that the application requires 1Mbps bandwidth on the requested paths. PS_2 is an ordered subset of PS_1 (possibly with some additional paths added in – see below) that meet the QoS requirements specified by the application. RS_2 contains the corresponding rating information for paths in the set PS_2. The rating tells an application how good a path is. The rating can be specific path information or a "rating score" calculated by NPAS. The QoS given by the application defines what metric to use when ranking and it determines the cutoff for that metric causing only a subset of the given paths to be returned. Ranking can be determined from a weighted combination of QoS metrics, and the QoS may define multiple dimensions to the QoS (e.g., min bandwidth of 1Mbps and max latency of 10ms). The weight of each metric can be determined based on the algorithm the application specified in the input. The algorithm that is used to rank path can be based on the input order of QoS metrics. For example, if the application specifies the bandwidth requirement before the latency requirement in the rating API, NPAS can order the paths that meet the requirements firstly by bandwidth and secondly by latency. Alternatively, the probability based algorithm such as the model described in thesis [80] can be used by NPAS to rank paths. Since a path that satisfies certain QoS requirements may not satisfy other QoS requirements, providing different QoS requirements will result in different ratings for the same path. With the first form of the rating API, an application can pick the paths in the order as recommended in path set PS_2.

Although the objective of the NPAS rating API is to rank the set of paths that the application gives, it is possible that NPAS knows of other paths (in addition to the ones presented by the application) that also work and meet the QoS requirements. The rating API may also recommend new paths that are not part of the set specified

by an application. An application can make its own decision on whether to use NPAS-recommended additional paths. So PS_2 may contain paths that are not in the requested set S_1.

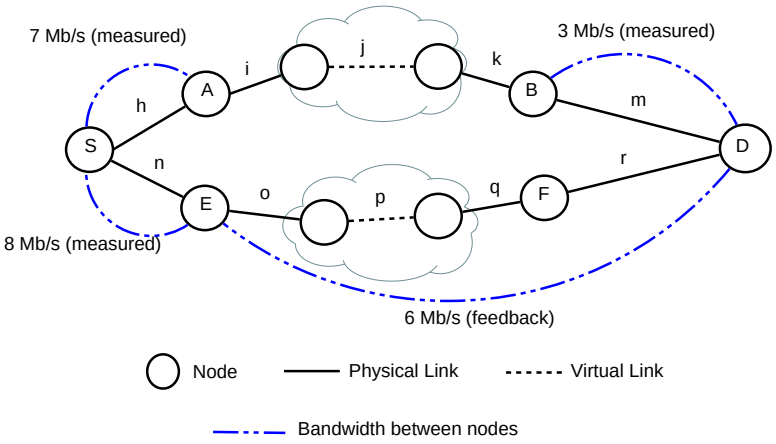

Figure 5.1: Using the Rating API (for Senders)

Figure 5.1 gives an example of how the rating API can help applications choose paths. Suppose a sender wants to choose a path to transfer files from node S to node D at a speed of 5Mbps. The sender uses the rating API to ask NPAS to rate paths *h-i-j-k-m* and *n-o-p-q-r*. The sender also informs NPAS that the bandwidth requirement is 5Mbps. NPAS then tries to evaluate path *h-i-j-k-m* and path *n-o-p-q-r*. NPAS only knows partial information about the paths. In particular, NPAS knows that the bandwidth of link h is 7Mbps and the bandwidth of link m is 3Mbps. NPAS does not know the bandwidth between node A and node B. However, NPAS can make a conclusion that path *h-i-j-k-m* does not meet the sender's requirements since the link between node B and node D has only 3Mbps bandwidth. When NPAS tries to evaluate path *n-o-p-q-r*, it realizes that the bandwidth between node E and node D has not been measured yet. But NPAS has received feedback from an application saying that the sending rate over path *r-q-p-o* can reach 6Mbps. Based on the feedback and the measured bandwidth of link n, NPAS concludes that path *n-o-p-q-r* meets the sender's requirements. To tell the sender how good a path is, NPAS needs to

calculate a rating score for both path *h-i-j-k-m* and path *n-o-p-q-r*. In this particular example, NPAS uses the following equation to calculate the rating score:

$$rating = (EstimatedBandwidth - RequestedBandwidth)/RequestedBandwidth.$$

This type of rating tells the sender that the estimated "head room" bandwidth of a path (i.e., Estimated - Requested) is a percent more/less than the requested bandwidth. Clearly one would like the extra "head room" (above the requested bandwidth) to be 0 or greater. Suppose NPAS uses the minimum bandwidth it knows about a path as the estimated bandwidth of that path. As a result, path *n-o-p-q-r* gets a rating score of $\frac{6-5}{5}$ and path *h-i-j-k-m* gets a rating score of $\frac{3-5}{5}$. A negative rating score means a path does not satisfy the application's requirements. The calculation of the rating score for path *h-i-j-k-m* does not take the bandwidth between node A and node B into account, and NPAS does not need to evaluate the path between node A and node B, because NPAS knows that path *h-i-j-k-m* does not satisfy the application's requirements. The rating of a path is calculated based on the current knowledge NPAS has about that path, which means that the rating will change as NPAS gets more up-to-date path information.

Note that receivers can also use the rating API. In particular, a receiver can ask NPAS to place constraints on incoming paths to the receiver. In other words, the rating API (for receivers) allows receivers to influence the paths that senders will take to reach them. For example, if a receiver has multiple interfaces, it may request that paths that come in over its primary interface be rated higher than paths that come in over the secondary interface when paths to primary interfaces are not congested. When NPAS gives path advice to the sender, the constraints from the receiver will be considered. To prevent spoofing of receivers, NPAS must verify the identity of the requester. This entails a client talking directly to the NPAS local request Server using an authentication protocol to prove the client has the right to define path preferences for itself. The path preferences are forwarded to the shared NPAS server by the NPAS

local request server and then can be distributed to other NPAS local request servers. In general, the rating API (for receivers) will be used to support servers that want to control their incoming traffic. The second form of the rating API (the rating API for receivers) that allows receivers to place constraints on paths is described as follows:

- Input: Incoming-Path, list of constraints
- Output: an error code

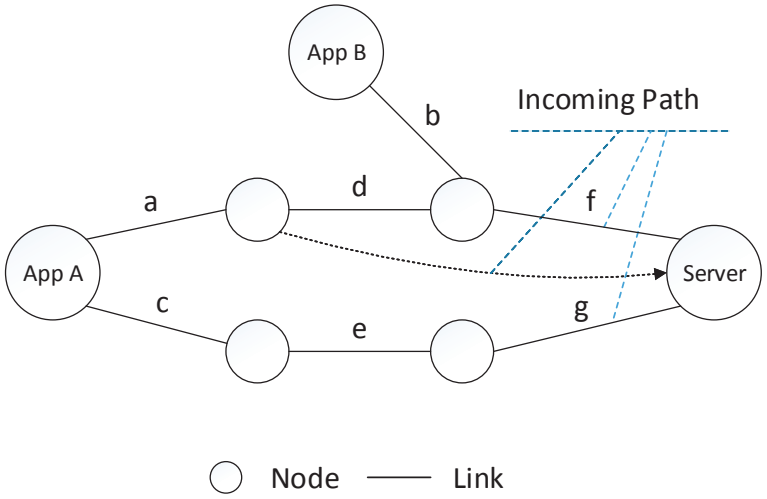

Figure 5.2: Using the Rating API (for Receivers)

With the second form of rating API, the receiver can specify constraints on an incoming path. The *Incoming-Path* in the API is a contiguous path that ends at the receiver. For example, in Figure 5.2, path *d-f*, path f, and path g are all incoming paths for the server. The constraint contains a *type*, a *match* and an *action*. The *type* of constraint can be bandwidth, flow, etc. Each type of constraint has its own way to specify the *match* where the match defines what conditions must be true for this constraint to be met (e.g., bandwidth greater than some value, flow matching some source/destination pair, etc). The *action* specifies whether the receiver accepts or rejects the traffic on the incoming path. The returned error code indicates whether

the constraints are accepted by NPAS. If the type is bandwidth, the match can be specified as the minimum amount of available bandwidth on the incoming link for the receiver to accept traffic coming from certain paths. For example, in Figure 5.2, the server can specify path *d-f* as the incoming path, bandwidth as the type, "minimum 5Mbps available bandwidth on link f" as the match, and "accept" as the action. This constraint tells NPAS that the server wants to accept traffic from paths that contain path *d-f* only if the available bandwidth on link f is greater than or equal to 5Mbps. Application A can use path *a-d-f* or path *c-e-g* to reach the server. However, if the available bandwidth on link f falls below 5Mbps, NPAS will recommend the path *c-e-g* to application A because of the constraint on path d-f. Application B can still use path *b-f* to reach the server since it does not use the incoming path *d-f*. If type of constraint is flow, the match contains information (e.g., port number, source address, destination address, etc.) to identify the flow. For example, the receiver can specify the port number 443 to identify "https" flows, and accept/reject "https" flows on different incoming paths. NAPS services can define their own supported constraints.

In addition to ranking paths, the rating API can also provide applications with detailed path and link information. Applications may not need NPAS to rank paths for them. With the detailed path and link information, an application can make its own decision about which path is best for the application. Suppose P_1 is a path or a link, T is the type of network metrics (e.g., bandwidth), V is the value of a network metric, TS is a time stamp, and P_2 represents a link (or a path) whose information is returned. The format of the third form of the rating API (the rating API for path and link information) is described as follows:

- Input: P_1, T
- Output: $[V, TS]$ or a list of $[P_2, V, T, TS]$

Using the third form of the rating API, an application can get the actual value V of a specific network metric T that is collected at the timestamp TS. P_1 can be a path that contains only a single link or multiple links. If P_1 contains multiple links, NPAS would return the link information it has for each link P_2 on the path P_1 as well as the information it has for the entire path. If the application does not specify the network metric T in the input, NPAS will return all the possible types of T (e.g., bandwidth, latency, jitter, loss rate, etc.). If information about multiple links (and/or multiple metrics) is returned, the output is specified as a list of P_2, V, T and TS.

5.2 The Scheduling API

Some applications may want to use several paths at the same time. In this case, the scheduling API is used to help an application spread its traffic over multiple paths. The format of the scheduling API is described as follows:

- Input: PS_1, Type, *Req*
- Output: PS_2, SS_2

In the scheduling API, an application provides the set of paths PS_1 it can use, the type of scheduling, and *Req* (the requirements of using these paths). NPAS supports two types of scheduling: "packet scheduling" and "flow scheduling". For "packet scheduling", *Req* contains the maximum number of paths that the application wants to use, the total throughput that the application wants to achieve, and may also contains information such as the maximum latency that the application allows. For "flow scheduling", *Req* contains the maximum number of paths that the application wants to use, the number of flows to send, and QoS requirements (the same QoS requirements as mentioned in the rating API) of each flow. The output contains the set of paths (PS_2) that are picked by NPAS, and the scheduling (SS_2) of traffic over the chosen paths. The scheduling SS_2 contains the desired sending rate on each

path in path set S_2 for "packet scheduling". For example, if an application provides 5 possible paths and wants to send 10Mbps traffic over 2 paths, the scheduling API may tell the application to send 3Mbps traffic on path 2, and 7Mbps traffic on path 3. For "flow scheduling", SS_2 contains a list of paths picked by NPAS, and each path is assigned one or more flows. For example, if an application provides 5 possible paths and wants to send 2 traffic flows with bandwidth requirements 3Mbps and 5Mbps respectively using 2 paths, the scheduling API may tell the application to send flow 1 on path 5, and flow 2 on path 1.

5.3 The Feedback API

The feedback API enables the application's ability to upload feedback. Feedback is important since it can be used by NPAS to give path advice when it is impossible to monitor the path directly or when the information obtained by monitoring links is not a good predictor of overall path performance. If all applications could be trusted to provide correct and reliable feedback, it would make sense to make the feedback interface available to all applications. However, we cannot assume that feedback is trustworthy. Without access controls, feedback can be misleading or lied about. Consequently, access to the feedback mechanism will be limited to nodes whose trustworthiness can be verified [1]. Therefore, to provide feedback, an application must provide its credentials along with the information about the path – including the time the path is evaluated by application. The path evaluation time helps NPAS determine whether the feedback is outdated, and the certificate can be used to verify the application.

Through the feedback API, an application can provide path measurement (V) for a specific type of metric (T), the time (TS) when the path is measured, and its certificate ($Cert$). In addition to the path statistics, an application can also inform

[1] In the future we would like to remove this constraint and instead use reputation-based systems or similar approaches to determine trustworthiness.

NPAS of its final choice (*Choice*) of routing paths after the path advice (*Advice*) was provided by NPAS. The feedback API is described as follows:

- Input: P, T, V, TS, $Cert$, [$Choice$, [$Advice$]],
- Output: a status code indicating whether the feedback is accepted by NPAS.

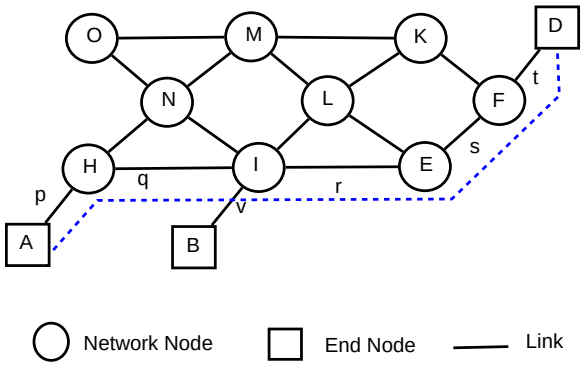

Figure 5.3: Using the Feedback API

Feedback can be rejected by NPAS due to various reasons, such as NPAS cannot verify an application using the application's certificate, the feedback is outdated, or the feedback is no longer needed by NPAS. The feedback API helps NPAS collect additional path information without taking additional measurements. Using the feedback API, an application can share its path statistics with NPAS, and make those statistics visible to other applications. For example, as shown in Figure 5.3, suppose a VoIP application making a voice call from node A to node D wants to provide feedback to NPAS. The VoIP application specifies the path it used is *p*-*q*-*r*-*s*-*t* (indicated by the blue dotted line) as the input to the feedback API. The VoIP application also sets the bandwidth metric (T) to 2Mbps (V). Having that feedback and information about link v, NPAS is able to rate path *v*-*r*-*s*-*t* for another VoIP application which wants to make a voice call from node B to node D.

In addition, the choice of routing paths that applications provided in the feedback can help NPAS recommend a path chosen by an application to another application

with similar requirements when NPAS does not collect sufficient measurement information to evaluate the path. The choice of routing paths can also tell NPAS how good the advice is. If the highest ranked path is not the path chosen by the application, it may imply that NPAS's advice is inaccurate, and thereby NPAS may need to collect the information on the related path more frequently. In short, the feedback can be used by NPAS in various ways to improve the quality of the NPAS service.

Chapter 6

Covering Paths

NPAS's goal is to be able to provide path recommendations to applications for the paths they are currently interested in. At any given time, applications are only interested in a subset of all possible paths in the Internet. Consequently, there is no need for NPAS to monitor paths that are not of interest to applications. In that sense, NPAS only needs to monitor enough of the Internet's links to "cover" (i.e., answer) the path requests it is receiving from applications.

To help us understand what types of path requests NPAS is likely to received, we analyzed existing network traffic traces to find out what sources and destinations were communicating. Given traffic traces along with a topology, we can compute the set of paths that a routing service would give to an application (which would, in turn, be given to NPAS to rank). Ideally, a monitoring system will want to monitor paths that many applications use, and leave paths which are used only by a small number of applications unmonitored or infrequently monitored to reduce the measurement overhead. However, applications that are in different locations (e.g., different ASes) tend to share only a part of path instead of the whole end-to-end path. Consequently, a shared monitoring system should only monitor shared links of a path, and let local monitoring systems monitor local links of paths. Both the shared monitor system and the local monitor system can work together to bring a more complete view of the end-to-end path information to applications. Individual ASes can have their own

monitoring mechanisms to help applications pick Intra-domain paths. Since paths often span multiple ISPs and ASes, NPAS focuses on the AS level topology and the AS level link/path.

To gain an understanding of how many links and nodes must be monitored to "cover" the typical network load generated by applications, we analyze a real-world traffic trace collected at the University of Kentucky (UKY). The trace shows the types of requests that NPAS can expect to receive from applications. Each record in the trace together with information about the Internet topology can be converted to a set of paths that represent a path request sent to NPAS.

6.1 Coverage Metrics

To measure how well a set of monitored links "cover" the set of path requests coming from applications, we need to define a set of "coverage metrics". First, we define a *path hit ratio* (Ph) to measure the percentage of paths that are "covered" in their entirety (i.e., all links along the path are being monitored). The path hit ratio is calculated as:

$$Ph = \frac{\sum_{f \in fset} p_f}{|fset|} \tag{6.1}$$

where $fset$ is the set of all traffic flows, p_f is either 0 or 1, indicating whether all the links along the paths being considered for use by flow f are being monitored, and $|fset|$ represents the number of flows. In short, the path hit ratio is the percentage of path requests (out of all path requests) where NPAS has complete information about the paths.

Second, we define the *link hit ratio* as the percentage of paths for which NPAS has collected information about some portion of the path. In other words, NPAS has *partial information* about a path – which could still be helpful to applications even

though it is incomplete information. We define the *link hit ratio* (Lh) as

$$Lh = \frac{\sum_{f \in fset} cn_f}{\sum_{f \in fset} tn_f} \tag{6.2}$$

where $fset$ is the set of all traffic flows, cn_f is the number of links that are monitored *and* are in the paths that flow f could use, and tn_f is the total number of links in the paths that flow f could use. The link hit ratio Lh indicates the average percentage of information a monitoring system can provide about a path when only a portion of the links are monitored.

Since both link hit ratio and path hit ratio indicate how much information NPAS can provide about a path, both link hit ratio and path hit ratio are used to measure the coverage that occurs when a particular set of links are selected as monitoring points.

6.2 What Paths Will Applications Request?

To compute path and link hit ratios, we need to understand the types of paths that NPAS will be asked for along with the frequency that those paths will be requested. In other words, we need to know something about the flows generated by end systems (i.e., their source, destination, and frequency). Given information about the destinations that applications in a particular domain (i.e., source AS) are trying to reach, the number of links needed to "cover" the majority of the paths to those destinations can be determined.

Using the UKY trace data described earlier, all the destinations in the trace and the number of flows destined to each of the destinations can be identified. The traffic trace consisted of NetFlow [29] records anonymized to protect user privacy. The trace contained approximately 200 million flows targeted at 10.4 million different IP addresses contained in 26,578 different ASes.

Destination ASes are sorted in descending order according to the number of flows whose destination IP addresses belong to the AS. Figure 6.1 describes the percentage

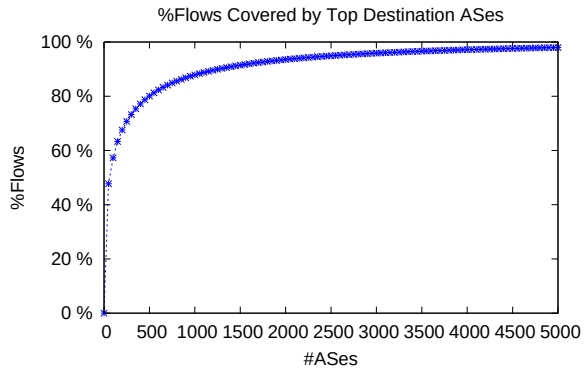

Figure 6.1: The Percentage of Flows that are Covered by Top Destination ASes

of flows that go to the top ASes. Despite the large number of destinations, we can see from Figure 6.1 that almost half of the flows were destined to the top 100 ("popular") ASes. Roughly 80% of the flows were destined to just 500 ASes, while about 90% of the flows were destined to the top 1000 ASes. This means that a large portion of the flow requests to NPAS (from a single AS) could be covered (handled) by monitoring a small subset of all paths in the Internet. Although an AS may send packets to a large number of ASes, monitoring paths to the top 500 "popular" ASes will help handle path requests for 80% of flows. Moreover, paths to unpopular ASes may share links with paths to the top 500 popular ASes. Consequently, monitoring paths to the top 500 popular ASes can also provide partial path information to applications destined for unpopular ASes.

Our study only looked at destinations originating from the University of Kentucky. However, NPAS needs to be able to respond to queries from any AS. If NPAS must monitor all paths from every source domain in the Internet to that domain's top 500 popular ASes, the number of paths to be monitored would explode. On the other hand, if many of the paths/links are shared across different source domains (i.e., two or more ASes use the same Internet links to reach their most popular destinations), then it may be possible to answer queries from multiple domains by only monitoring a small number of paths/links.

To emulate traffic originating from multiple ASes, we again use the UKY flow trace. Although the same set of destinations is used, the flows are re-originated from a variety of different ASes (i.e., not UKY). In other words, the UKY AS in the flow trace is replaced with other ASes including MIT, Columbia, Purdue, Utah, UCB, and FSU. This set of source ASes is later expanded to include ASes all across the world. We use the UKY trace from these other domains, because the UKY trace is believed to be representative of many other institutions' traffic patterns. The UKY trace, like most other academic institutions' traffic, is dominated by accesses to widely popular destinations such as Google, Facebook [81], NetFlix [82], and YouTube [83]. Although the specific flow distribution may differ slightly, the same set of destinations are also widely popular at other institutions, and tend to dominate their accesses as well.

6.3 How Many Links Should NPAS Monitor?

To determine how many links NPAS needs to monitor in order to achieve a high path/link hit ratio, we used the traffic loads from the ASes described above to generate path requests to NPAS and then measured the path/link hit ratios that result from monitoring an increasing number of the most popular links.

6.3.1 Single-Source Monitoring

We began by picking the University of Kentucky AS (UKY AS) as an example AS for our single source monitoring analysis. The number of links in the paths to the top ASes from UKY grows approximately linearly with the number of ASes included in the "top" group. The paths to the top 500 ASes contain about 600 links. As shown below, monitoring 600 links can cover a relatively high percentage of flows (e.g., about 80% flows) for a single AS.

Figure 6.2 shows how the link hit ratio grows with the number of links being monitored. Any given link may be used by multiple flows. In this case, links that are

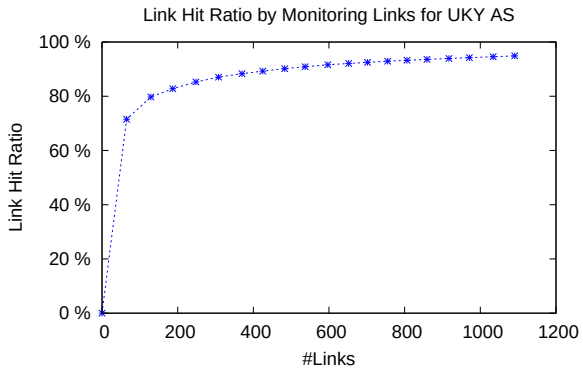

Figure 6.2: The Link Hit Ratio vs the Number of Links Monitored for UKY

part of paths to top ASes are monitored since these links may have potential to be used by more flows than the links in the paths to unpopular ASes. In Figure 6.2, the number of ASes that applications want NPAS to be able to answer questions about is increased, which in turn increase the number of links that must be monitored to ensure a high link hit ratio. From Figure 6.2, we can see that a link hit ratio of more than 60% can be achieved by monitoring as few as 100 links, and the link hit ratio reaches around 90% by monitoring approximately 600 links.

6.3.2 Identifying Popular Links for Multiple Sources

A link hit ratio of around 90% can be reached for the UKY AS when about 600 links are monitored. However, NPAS needs to be able to respond to queries from any source AS.

Having identified the popular destinations, any link on a path leading from a source AS to any of the popular destination ASes (e.g., the top 500 popular ASes) becomes a candidate link to be monitored. To maximize path coverage while at the same time minimizing the number of links to be monitored – links needs to be picked according to their contribution to the link hit ratio and/or the path hit ratio.

We defined a single *score* that combines both ratios into a single metric. In particular, LR_l and PR_l are used to identify a link l's contribution to the overall

link hit ratio and the overall path hit ratio respectively. LR_l and PR_l of a link l are calculated using equation 6.3, where NF_T is the total number of popular flows (i.e., flows destined for popular ASes), NF_l is the number of popular flows that involve link l (i.e., the number of flows destined for popular ASes that use link l), and NP_l is the number of popular flows whose paths will be completely monitored if the link l is added to the current set of links being monitored.

$$LR_l = \frac{NF_l}{NF_T} \quad and \quad PR_l = \frac{NP_l}{NF_T} \tag{6.3}$$

Note that NP_l depends on the current set of links being monitored. Even if a path has all but one of its links in the set of links being monitored, the NP_l value for all links included so far will be zero until the last missing link is added. In other words, as more links are added to the set of links being monitored, more and more links will suddenly find themselves contributing to the path hit ratio (i.e., their PR_l value will increase by adding other links to the monitored set). Consequently, when deciding to add a link to the monitored set or not, we introduce a third factor, FPR_l, that indicates a link's "potential" to contribute in the future. A node's *future path hit ratio* contribution, FPR_l, is based on the number of paths that l has the potential to help (should all the other links along those paths be selected). Specifically FPR_l is computed using equation 6.4, where $NPath_T$ is the total number of paths that are used by flows destined for popular ASes, and $NPath_l$ is the number of paths in $NPath_T$ that link l is a part of.

$$FPR_l = \frac{NPath_l}{NPath_T} \tag{6.4}$$

A *link score (LS)* can now be computed using equation 6.5.

$$LS = \alpha FPR_l + \beta LR_l + \gamma PR_l \tag{6.5}$$

The link score LS is a weighted average, where α, β, and γ are weights for these three factors. Being able to answer queries about entire paths is very important, so

we want to monitor links that will contribute to a complete path the most quickly. Thus we give the most weight to α. We give more weight to α than to γ because high γ can lead to suboptimal results since PR_l is calculated based on the current set of links being monitored. In addition, we want to give a heavier weight to β than γ in order to pick links that are used by many flows. Given these goals along with simulations we performed to evaluate the parameter space, we ended up selecting values of $\alpha = 0.8$, $\beta = 0.15$, and $\gamma = 0.05$ as values that resulted in the best link and path hit ratios.

Using the trace data and the topology graph from CAIDA [84] as input, our algorithm to pick shared links to monitor is shown below:

1. Calculate LS scores of all candidate links.
2. Update PR_l as links are added to the set of links to be monitored.
3. Update LS scores of candidate links that are affected by changing PR_l.
4. Pick the link with the highest LS score to monitor, and move the link from the candidate link set to the "to be monitored" link set.
5. Repeat: goto step 2 until the number of links picked reaches the predetermined limit or no more candidate links are left.

6.3.3 Global (Shared) Monitoring

Using the previously described algorithm to select (beneficial) links to monitor, the link hit ratio that various ASes would experience can then be computed. In Figure 6.3, we pick 7 University ASes that are geographically apart from each other in the United State as our representative ASes to evaluate whether our strategically picked links can achieve a high link hit ratio for traffic originating from different locations. The links to be monitored are picked globally based on all source ASes (not just the representative ASes) in the topology. We assume all source ASes have a traffic pattern similar to

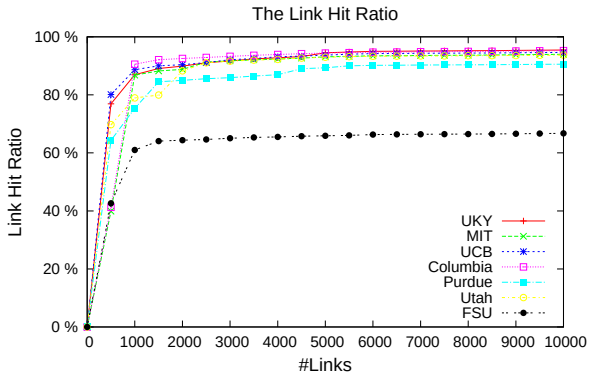

Figure 6.3: The Link Hit Ratio by Monitoring Top Globally Shared Links

the UKY AS in this experiment. We can see from Figure 6.3, the link hit ratio grows with the number of links being monitored. By monitoring 5000 important links, we can achieve an 80% to 95% link hit ratio for the traffic flows originating from 6 out of 7 ASes. However, monitoring the same set of 5000 links results in a relatively low link hit ratio for the Florida State University (FSU) AS, because some heavily used local links coming out of the FSU AS are only shared by a few ASes and thus are not monitored globally. However, as we will see in Section 6.3.4, if there are local monitoring systems that can monitor links which are not monitored by the shared monitoring system, but yet are heavily used by a specific AS, the link hit ratio can be improved for that AS.

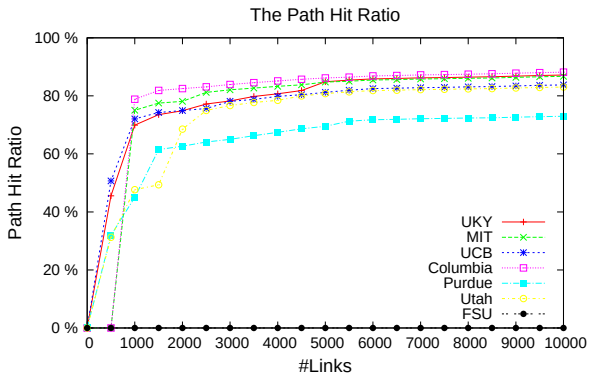

Figure 6.4: The Path Hit Ratio by Monitoring Top Globally Shared Links

Figure 6.4 shows how the path hit ratio grows with the number of globally monitored important links. The same set of representative ASes that was used in Figure 6.3 are evaluated in Figure 6.4. We can see that when 5000 shared links are monitored, about 75% to 85% of the paths originating from UKY, MIT, UCB, Columbia and Utah are completely covered by those links. The FSU AS has a path hit ratio of 0% since key local links are not being monitored. Although monitoring 5000 shared links globally can provide end-to-end path information for a high percentage of traffic for some ASes (e.g., the Columbia AS), local links may need to be monitored to provide end-to-end path information for traffic originating from certain ASes (e.g., the FSU AS).

6.3.4 Local (Private) Monitoring

Since the shared monitoring system may not want to monitor links that are only used by traffic originating from a specific AS, a local monitoring system can be deployed for each AS to monitor local links that are directly attached to the AS. The local monitoring system only needs to monitor local links that are on the paths to popular ASes (e.g., the top 500 ASes) and are not being monitored globally. The local link information is kept locally, and the NPAS local request server provides path advice to local applications using both the shared link information distributed by the shared NPAS server and local link information from the local AS.

Figure 6.5 and Figure 6.6 show that incorporating local monitor system information greatly improves both the link hit ratio and path hit ratio. In the experiment, when NPAS monitors 5000 shared links, the local monitoring system only needs to monitor 1 local link for FSU and UCB, 3 local links for MIT, and no local link for other representative ASes. If we compare Figure 6.5 with Figure 6.3 and Figure 6.6 with Figure 6.4, we can see that monitoring a few local links helps improve both the link hit ratio and the path hit ratio, especially for the FSU AS which went from 0%

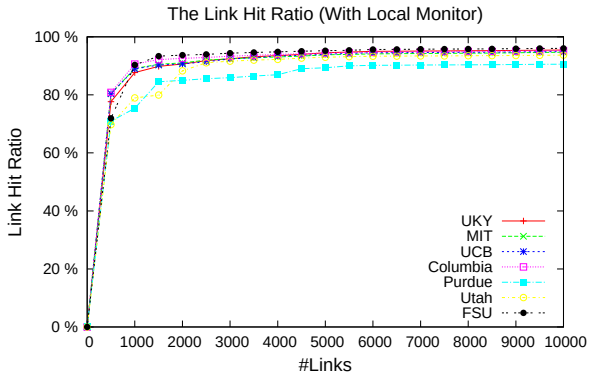

Figure 6.5: The Link Hit Ratio with Local Monitor

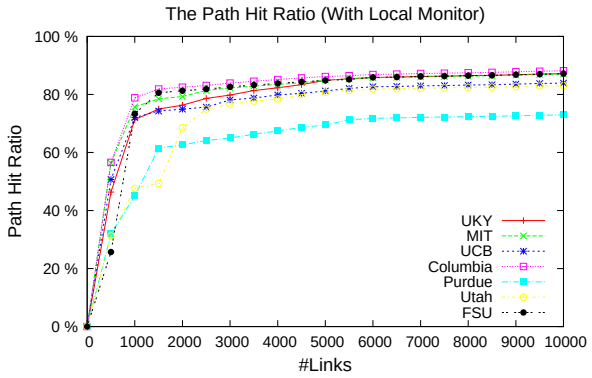

Figure 6.6: The Path Hit Ratio with Local Monitor

to 80% for the path hit ratio. We can see that by monitoring 5000 shared links and a few local links, the monitoring system is capable of providing applications with around 90% link information and around 60% to 85% path information for these representative ASes.

6.3.5 Multi-source Multi-route Monitoring

In the future Internet, applications will have the ability to choose among multiple routing paths. To evaluate whether our strategically picked links can still provide a high link/path hit ratio in a network that supports multiple routing paths, five paths are picked from each representative AS to each possible destination AS. These

five paths are picked based on shortest path criteria, but they are distinct (i.e., non-overlapping) from each other.

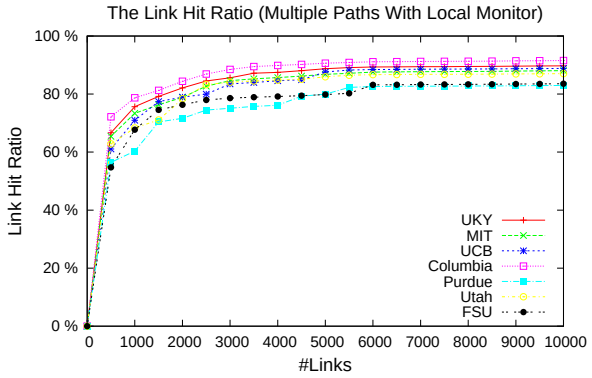

Figure 6.7: The Link Hit Ratio with Local Monitor and Multiple Paths

Figure 6.7 shows the link hit ratio for representative ASes when five possible paths can be used by applications. We can see that the link hit ratio for five possible routing paths is only slightly lower than the link hit ratio for a single routing path, and monitoring 5000 links can still provide a high link hit ratio for five possible routing paths.

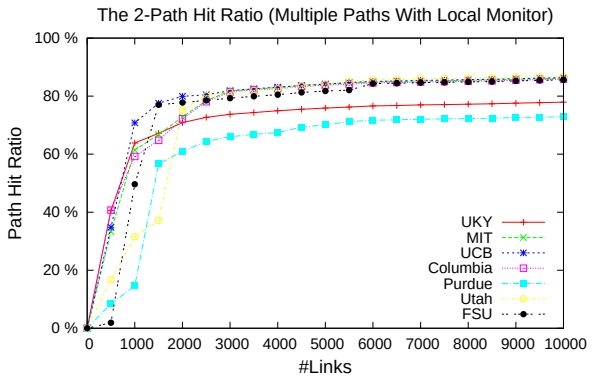

Figure 6.8: The 2-Path Hit Ratio with Local Monitor and Multiple Paths

Figure 6.8 shows the path hit ratio when 2 or more paths are covered by the links being monitored. More specifically, a flow (request) is considered to contribute to the 2-path hit ratio when at least 2 possible paths (out of 5 paths) are completely

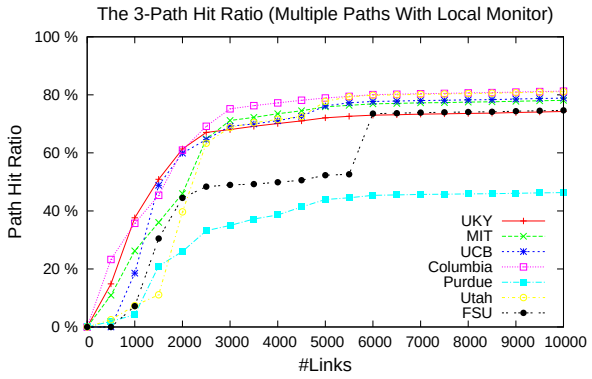

Figure 6.9: The 3-Path Hit Ratio with Local Monitor and Multiple Paths

monitored by NPAS. Figure 6.9 shows the path hit ratio when 3 or more paths are monitored. From Figure 6.8 and Figure 6.9, we know that monitoring 5000 links and a few local links can still provide approximately a 75% path hit ratio for 2 and 3 paths, which implies that around 75% traffic can get end-to-end path information for at least 2 to 3 paths. In this example, one AS (the Purdue AS) has a below average 3-path hit ratio because some of links that are used by the AS are not popular enough to be monitored globally. However, the Purdue AS has a high link hit ratio, which means that applications can still get partial path information for paths that are not completely covered by links being monitored.

6.3.6 Future Traffic Monitoring

We expect that top ASes and candidate shared links do not change frequently, and links that are chosen based on historical traffic can still have good path/link coverage for future traffic. To prove our expectation, the UKY traffic trace is split into two parts: each part presents traffic traces for half of a day. The data of the first half day is used to find the top ASes and links to be monitored, and the data of the second half day is used to find out how future flows will be covered by the links picked based on the historical data.

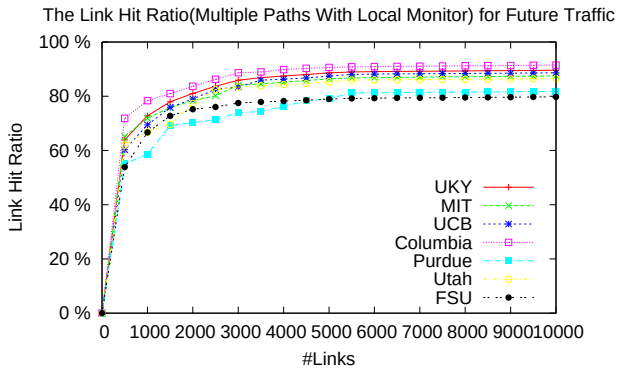

Figure 6.10: The Link Hit Ratio for Future Traffic with Local Monitor

Figure 6.10 shows the link hit ratio of the flows in the second half day when the links to be monitored are calculated based on the flows in the first half day. By comparing Figure 6.10 and Figure 6.7 , we know that when the links being monitored are chosen based on the previous 12 hours' traffic, the overall link hit ratio is still close to the link hit ratio by monitoring links that are chosen based on the current traffic. Therefore, our method of picking links to monitor based on recent traffic works for the real world traffic. However, an unpopular AS may become popular over time. In Section 7.1, we talks about how NPAS can keep track of ASes that have recently become popular.

6.3.7 Monitoring for International ASes

Even though our strategically picked links lead to high link and path hit ratio for representative ASes, these ASes are all located in the United States. To prove that monitoring our strategically picked links also provides a good link/path hit ratio for ASes outside the United States, we pick a second set of representative University ASes, including Tsinghua AS (Tsinghua University in China), Oxford AS (University of Oxford in UK), USP AS (University of Sao Paulo in Brazil), MSU AS (Moscow State University in Russia), UniMelb AS (University of Melbourne in Australia), and UCT AS (University of Cape Town in South Africa).

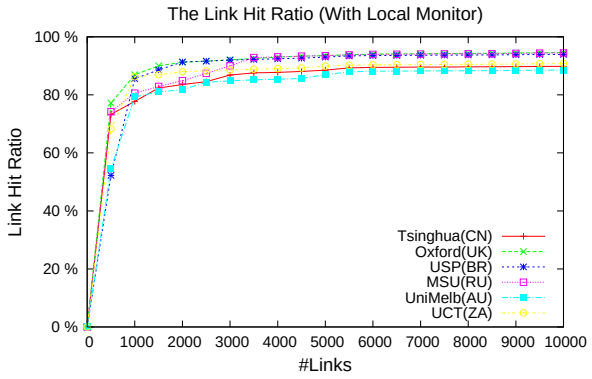

Figure 6.11: The Link Hit Ratio with Local Monitor for International ASes

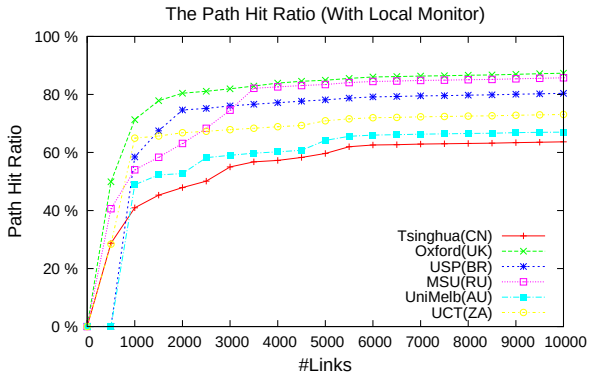

Figure 6.12: The Path Hit Ratio with Local Monitor for International ASes

Figure 6.11 and Figure 6.12 show the link hit ratio and path hit ratio by monitoring the same set of strategically picked links for the second set of ASes. We can see from the figures that by monitoring 5000 links, the link hit ratio for the second set of ASes reaches around 80% to 90%, and the path hit ratio varied from 60% to 85%. Compared to the representative ASes within the United States, the second set of ASes achieve similar link hit ratios and a slightly lower path hit ratios.

Links to be monitored are not strategically picked based on special ASes, rather links are picked based on how links are shared for all ASes. Therefore, monitoring the same set of 5000 links can also achieve a good path hit ratio for ASes outside the US.

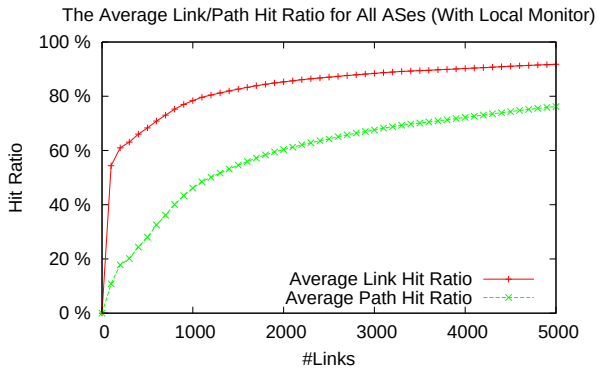

Figure 6.13: The Average AS Link/Path Hit Statistics with Local Monitor

Figure 6.13 shows the average link/path hit ratio of all ASes (around 40000 ASes in the sample AS topology) as the number of links monitored increases. To learn the overall link/path hit ratio for all ASes in the AS topology, we calculate the link/path hit ratio for each AS by monitoring a certain number of links, and then calculate the average link/path hit ratio of all ASes. As shown in Figure 6.13, we can achieve an average link hit ratio of 91% and an average path hit ratio of 76% for all ASes by only monitoring 5000 links globally with local monitor applied to fill in missing information.

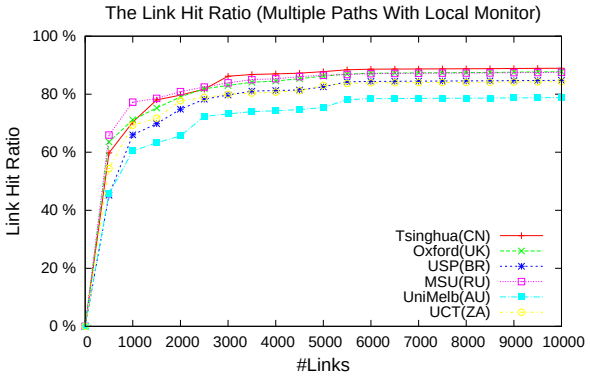

Figure 6.14: The Link Hit Ratio with Local Monitor and Multiple Paths for International ASes

Figure 6.14 and Figure 6.15 show the link hit ratio and 2-path hit ratio when 5 possible paths are used for each source and destination pair. From Figure 6.14 and

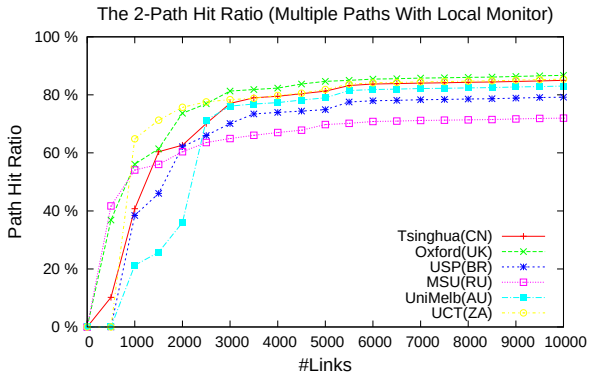

Figure 6.15: The 2-Path Hit Ratio with Local Monitor and Multiple Paths for International ASes

Figure 6.15, we know that these international ASes still achieve around 75% to 85% link hit ratios and around 70% to 85% 2-path hit ratios.

In short, we are able to identify a small number of links to monitor to achieve a high link/path hit ratio. Monitoring our strategically picked links has the potential to help the majority of traffic across the Internet choose routing paths.

6.3.8 Monitoring for Multiple Traffic Patterns

Assuming that traffic from all source ASes has the same destination distribution pattern as the traffic from UKY ASes may not provide accurate simulation for traffic originated from non-university ASes. To demonstrate that our method for picking monitoring links will also lead to high path coverage when ASes exhibit different traffic patterns than university traffic patterns, we classify ASes into categories and assigned distinct traffic distribution patterns. Based on the AS registration information, we separate ASes into *university* ASes, *corporation* ASes, *government* ASes, *ISP* ASes, and *unclassified* ASes.

According to the global Internet phenomena report [85], several popular services (e.g., Netflix [82], Youtube [83], Facebook [81] etc.) – which we will call *streaming and social media services* – are responsible for more than 50% of the Internet traffic. Based

on distinct usage patterns for users from various types of ASes, we vary the traffic distribution to these popular services for different ASes. For example, the percentage in the traffic report [85] (e.g., Netflix(30%), Youtube(17%), Facebook(4%), Google Play(3%), iTunes(2%), etc.) can be used as the traffic distribution to streaming and social media services for the unclassified ASes. Based on our recent study of UKY traffic, traffic to a popular service like Netflix can account for more than 50% of the UKY traffic. Therefore, for the university ASes, we increase the percentage of traffic (e.g., a total of 65%) to streaming and social media services. For government ASes and corporation ASes, we expect they have a much lower percentage of traffic that goes to sites like Netflix, and so we reduce the percentage of traffic (e.g., a total of 20%) to streaming and social media services. Since personal Internet use at home is mostly for entertainment purpose and personal traffic is carried through the ISP ASes (i.e., Cable Company ASes), we expect the ISP ASes may have a higher percentage of traffic (e.g., 75%) to streaming and social media services. For traffic designated at unpopular services, the UKY traffic distribution is used.

Table 6.1: Traffic to Streaming and Social Media Services

Source AS	Streaming and Social Media Services
University	65%
Corporation	20%
Government	20%
ISP	75%
Unclassified	59%

With distinct traffic distribution patterns to streaming and social Media Services for various ASes as described in Table 6.1, we reran our algorithm to pick links to be monitored and evaluate the path/link hit ratio for different ASes. Figure 6.16 shows the average link/path hit ratio for all ASes in our AS topology. The links are picked to be monitored based on traffic from all source ASes. The link/path hit ratio of an AS is calculated based on the AS's traffic distribution. We can see from Figure 6.16

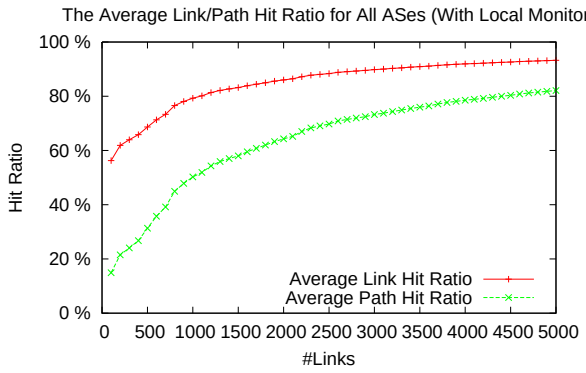

Figure 6.16: The AS Link/Path Hit Statistics with Local Monitor and Multiple Traffic Patterns

that the link hit ratio reaches around 93% and the path hit ratio reaches around 82% when 5000 links are monitored globally with local monitor.

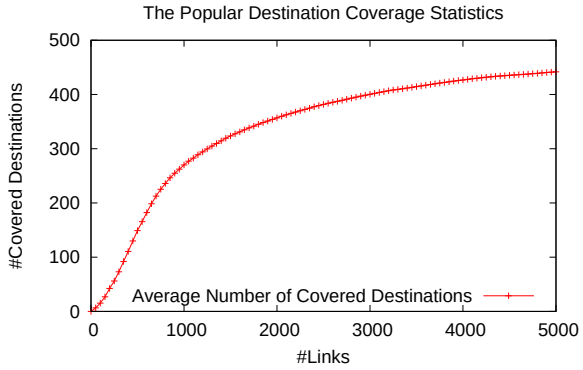

Figure 6.17: The Top 500 ASes Coverage Statistics

Figure 6.17 shows the coverage of the top 500 ASes by monitoring various number of links. From Figure 6.17, we learn that monitoring 5000 shared links and local links can cover an average of roughly 440 out of 500 popular destination ASes for all source ASes. Since most of the popular destinations can be covered, our strategically picked links can provide a high link/path hit ratio in general despite the fact that distinct source ASes may have distinct traffic distribution patterns to the popular destination ASes.

6.4 Selecting Monitoring Points

We have learned how to select shared links to monitor that will "cover" a large percentage of the flows originating from multiple source domains. The next challenge is to enable monitoring of these links at the ASes to which they are connected. "Enabling monitoring" does not mean deploying new monitoring tools in the monitoring points. Rather, it means to request that the corresponding ASes send measurements of certain links. The monitoring points will typically be controlled by individual ASes, and ASes can use their existing monitoring infrastructure to collect path information that NPAS requests.

As shown in the previous section, only 5000 shared links need to be monitored to "cover" the Internet. Using the AS topology graph, we can see that the top 5000 shared links are attached to roughly 3000 ASes. In this case, NPAS would receive path requests from which NPAS could extract the 5000 most frequently occurring links. Given the 5000 links, NPAS would then enable monitoring at the 3000 ASes that are connected to the ends of each link. Ideally, NPAS can get information from both sides the link (i.e., the ASes on either side of the link).

While it may be reasonable to collect data from 3000 ASes, it may be unfeasible or impractical to manage relationships with 3000 ASes (setting up/managing SLAs, authentication mechanisms, authorization policies, etc). Consequently, there are reasons to limit the number of ASes NPAS must monitor, even if it means reducing NPAS's ability to respond to path requests. Ideally NPAS would like to minimize the number of ASes it needs to communicate with and yet maximize the number of path requests that NPAS can provide information about.

In cases where the path metric requested is bandwidth or hop-by-hop round trip delays (as opposed to one-way delays), it is possible to only monitor one end of a link, further reducing the number of ASes NPAS needs to interact with. In the following, we describe how NPAS selects ASes to monitor. We call these ASes *Monitoring Points*

(MPs).

6.4.1 One AS per Link Monitoring

There are around 3000 ASes that the top 5000 shared links are attached to. However, only a small portion of these ASes need to be monitored in order to cover a majority of the top 5000 shared links (assuming NPAS only monitors one AS per link). To achieve maximal coverage but yet limit the number of ASes NPAS needs to interact with if NPAS only needs to monitor one end of a link, NPAS uses the link score computation from equation 6.5 to select links to be monitored – and (indirectly) to select ASes where monitoring is to be enabled. In particular NPAS:

1. Begins by marking all ASes and links as unmonitored.

2. It then calculates the link score of every link using equation 6.5. However, to emphasize the contribution of the link hit ratio and deemphasize the (highly varying) path hit ratio, NPAS sets $\alpha = 0.1$, $\beta = 0.9$, and $\gamma = 0$. This ensures that ASes containing links that are heavily used by applications will be picked first.

3. It then picks an unmonitored AS that will maximize the total link score of all monitored links, and marks the AS and links that are directly attached to that AS as being monitored.

4. Repeat the process (going to step 3) until the maximum allowed number of ASes have been selected.

We use the above algorithm to select one AS per link (noting that some links may still end up having both ends monitored). In each iteration of the algorithm, we try to find an AS that contributes most to the link score. The relationship between the number of top 5000 shared links that can be monitored and the number of ASes that NPAS chooses to monitor based on the above algorithm is shown in Table 6.2. We

Table 6.2: The Top 5000 Link Coverage from Monitoring ASes (One AS per Link)

#Monitoring_Points	#Links_Covered	#Links_NotCovered
50	4249	751
100	4472	528
150	4633	367
200	4735	265
250	4820	180
300	4872	128
350	4922	78
400	4972	28
428	5000	0

can see from the Table 6.2, the first 50 added ASes can cover 4249 out of 5000 links and the first 100 added ASes can cover upwards of 4400 links.

Table 6.3: The Link/Path Hit Ratio for Monitoring 50 ASes (One AS per Link)

	Without Local Links		With Local Links		
	%Link Hit	%Path Hit	%Link Hit	%Path Hit	#Links(#local_links)
UKY	92.25%	79.24%	92.99%	80.73%	4249(1)
MIT	91.48%	81.50%	92.18%	81.50%	4249(3)
UCB	92.30%	77.88%	92.58%	77.88%	4249(1)
Columbia	92.82%	82.47%	92.84%	82.47%	4249(0)
Purdue	56.89%	0.00%	86.75%	68.56%	4249(2)
Utah	63.69%	0.00%	91.74%	77.26%	4249(1)
FSU	64.55%	0.00%	93.85%	81.53%	4249(1)

Table 6.4: The Link/Path Hit Ratio for Monitoring 100 ASes (One AS per Link)

	Without Local Links		With Local Links		
	%Link Hit	%Path Hit	%Link Hit	%Path Hit	#Links(#local_links)
UKY	93.88%	83.15%	93.89%	83.15%	4472(0)
MIT	92.49%	83.72%	93.18%	83.81%	4472(3)
UCB	93.28%	80.21%	93.55%	80.21%	4472(1)
Columbia	93.81%	84.88%	93.83%	84.88%	4472(0)
Purdue	82.10%	65.83%	88.72%	70.14%	4472(1)
Utah	92.46%	79.48%	92.46%	79.48%	4472(0)
FSU	65.28%	0.00%	94.58%	83.81%	4472(1)

Table 6.3 shows that the link hit ratio and the path hit ratio (using the UKY traffic distribution pattern) for traffic originating from some source ASes when 50 ASes are monitored. Seven university ASes that are geographically located apart

from each other in the United States are used as our first set of example ASes. We can see that example ASes in the table can exceed a link hit ratio of 85% and exceed a path hit ratio of 68% by also monitoring local links. When 100 ASes are monitored, the link hit ratio can reach 90% and the path hit ratio can reach above 70% for these example ASes (shown in Table 6.4). In short, if NPAS only needs to monitor one AS per link, NPAS can provide link/path information for the majority of Internet traffic by monitoring 50 to 100 ASes.

6.4.2 Two ASes per Link Monitoring

As noted earlier, both sides of a link need to be monitored in order to accurately measure certain network information (e.g., one-way delay). In such cases, NPAS can identify ASes/links to be monitored using the following steps:

1. Mark all ASes and links as unmonitored.
2. Calculate the link score of every link using equation 6.5 (using $\alpha = 0.1$, $\beta = 0.9$, and $\gamma = 0$).
3. Pick the first 50 ASes according to the method that is used previously to pick one AS per link, and mark these 50 ASes as monitored.
4. Calculate the AS score of unmonitored ASes. An AS A's score is calculated as the sum of link score of links that connect the current unmonitored AS A and any other AS that is marked as monitored.
5. Mark the AS with maximum AS score as monitored.
6. Repeat the process (going to step 4) until the maximum allowed number of ASes have been selected.
7. Order the candidate links according to the algorithm described in Section 6.3.2.

8. Based on the link order, mark the link as monitored if both endpoint ASes of the link are marked as monitored until the maximum allowed number of Links have been selected or no more such links exist.

Table 6.5: The Link/Path Hit Ratio for Monitoring 300 ASes (Monitoring Both Endpoints)

	Without Local Links		With Local Links		
	%Link Hit	%Path Hit	%Link Hit	%Path Hit	#Links(#local_links)
UKY	86.31%	66.36%	86.31%	66.36%	3955(0)
MIT	44.90%	0.00%	85.37%	67.93%	3955(6)
UCB	88.51%	67.65%	88.79%	67.65%	3955(1)
Columbia	46.02%	0.00%	85.64%	67.99%	3955(2)
Purdue	52.12%	0.00%	81.98%	53.50%	3955(2)
Utah	32.50%	0.00%	60.56%	0.00%	3955(1)
FSU	59.97%	0.00%	89.27%	67.96%	3955(1)

Table 6.6: The Link/Path Hit Ratio for Monitoring 600 ASes (Monitoring Both Endpoints)

	Without Local Links		With Local Links		
	%Link Hit	%Path Hit	%Link Hit	%Path Hit	#Links(#local_links)
UKY	91.09%	77.91%	91.09%	77.91%	5000(0)
MIT	90.61%	79.27%	90.66%	79.27%	5000(1)
UCB	92.24%	78.22%	92.52%	78.22%	5000(1)
Columbia	90.85%	79.31%	90.87%	79.31%	5000(0)
Purdue	55.84%	0.00%	85.70%	61.71%	5000(2)
Utah	90.97%	75.32%	90.97%	75.32%	5000(0)
FSU	63.79%	0.00%	93.10%	79.18%	5000(1)

Table 6.5 and 6.6 describe the link hit ratio and the path hit ratio by monitoring 300 ASes and 600 ASes if both sides of a link are monitored. As the number of ASes being monitored reaches 600, the global monitoring system can achieve about 90% link hit ratio and 70% path hit ratio for the sample ASes with local monitor. The maximum number of shared links to be monitored is set to 5000, and the 5000 links that are monitored in Table 6.6 contain additional candidate links that are ordered after the 5000th popular links since not all the first 5000 popular links are covered by these 600 ASes.

6.4.3 Monitoring Point Selection for Multi-route Monitoring

We have demonstrated that our proposed method of picking monitoring points can achieve a high link hit ratio and path hit ratio when a single routing path is used between a source and destination. If a network supports multiple routing paths, learning the path conditions about all possible paths is important to help applications decide the final routing paths. In the following, we study the link and path hit ratio for our strategically picked monitoring points in a network where the routing service provides five possible paths to applications. Table 6.7 shows the link and path hit ratio when 100 ASes are monitored if only one side of a link needs to be monitored. We can see that the link hit ratio for all five possible paths (Table 6.7) is only slightly lower than the link hit ratio for the case of a single routing path (Table 6.4). In addition, a 70% to 80% 2-path/3-path hit ratio can be achieved.

Table 6.8 shows the link/path hit ratio when 600 ASes are picked if we monitor both sides of a link. From Table 6.8, we can see that monitoring 600 ASes can still achieve a high link hit ratio and cover at least 2-3 paths for around 70% to 80% of the traffic flows from the most representative ASes.

Table 6.7: The Link/Path Hit Ratio for Monitoring 100 ASes with Multiple Paths (Monitoring One Endpoint)

	With Local Links				
	%Link Hit	%1-Path Hit	%2-Path Hit	%3-Path Hit	#Links(#local_links)
UKY	88.28%	84.85%	74.54%	70.55%	4472(0)
MIT	86.28%	85.27%	82.76%	77.98%	4472(3)
UCB	87.40%	85.40%	83.08%	75.15%	4472(1)
Columbia	89.85%	85.23%	82.45%	77.06%	4472(0)
Purdue	78.61%	83.00%	73.42%	42.48%	4472(1)
Utah	85.54%	85.26%	82.80%	76.47%	4472(0)
FSU	79.33%	85.44%	80.59%	50.95%	4472(1)

6.4.4 Monitoring Point Selection for International ASes

After the evaluation of the first set of example ASes, we evaluate the link/path hit ratio for the second set of example ASes that are located outside the United States.

Table 6.8: The Link/Path Hit Ratio for Monitoring 600 ASes with Multiple Paths (Monitoring Both Endpoints)

	With Local Links				
	%Link Hit	%1-Path Hit	%2-Path Hit	%3-Path Hit	#Links(#local_links)
UKY	88.36%	79.08%	78.29%	77.32%	5000(0)
MIT	88.77%	79.31%	79.29%	78.22%	5000(1)
UCB	88.89%	79.29%	79.14%	78.26%	5000(1)
Columbia	90.53%	79.31%	79.18%	78.28%	5000(0)
Purdue	76.03%	76.79%	48.07%	30.35%	5000(2)
Utah	87.07%	79.31%	79.30%	78.17%	5000(0)
FSU	87.03%	79.33%	78.80%	76.52%	5000(1)

Table 6.9: The Link/Path Hit Ratio for Monitoring 100 ASes with Multiple Paths (Monitoring One Endpoint)

	With Local Links				
	%Link Hit	%1-Path Hit	%2-Path Hit	%3-Path Hit	#Links(#local_links)
Tsinghua(CN)	87.63%	85.01%	81.96%	70.97%	4472(1)
Oxford(UK)	86.10%	85.09%	83.41%	74.51%	4472(16)
USP(BR)	81.48%	79.63%	73.96%	55.37%	4472(5)
MSU(RU)	85.84%	84.46%	68.78%	64.55%	4472(6)
UniMelb(AU)	74.93%	84.94%	77.78%	38.09%	4472(1)
UCT(ZA)	81.15%	84.20%	80.29%	64.16%	4472(4)

By monitoring 100 ASes – one AS per link (as shown in Table 6.9) – the link hit ratio for most example ASes is around 80%, and the 2-path hit ratio for most ASes is about 70% to 80%. Table 6.10 shows the link/path hit ratio for international ASes if NPAS monitors both sides of a link. We can see from Table 6.10, most ASes achieve high link hit ratios and high 2-path/3-path path hit ratios.

Table 6.10: The Link/Path Hit Ratio for Monitoring 600 ASes with Multiple Paths (Monitoring Both Endpoints)

	With Local Links				
	%Link Hit	%1-Path Hit	%2-Path Hit	%3-Path Hit	#Links(#local_links)
Tsinghua(CN)	92.43%	79.32%	79.24%	78.38%	5000(1)
Oxford(UK)	88.64%	79.35%	79.29%	79.15%	5000(7)
USP(BR)	83.46%	79.24%	77.76%	69.09%	5000(5)
MSU(RU)	89.18%	79.31%	79.12%	78.17%	5000(6)
UniMelb(AU)	88.18%	79.32%	79.16%	77.97%	5000(1)
UCT(ZA)	87.19%	79.27%	78.45%	76.41%	5000(4)

6.5 Network Overhead of Covering Popular Paths

We have shown that NPAS only needs to monitor 5000 popular links and 100 to 600 ASes in order to cover around 75% traffic. Collecting and distributing path measurements consume network bandwidth. To design a scalable NPAS, the potential network overhead must be kept at a reasonable level.

A working NPAS system involves the following procedures:

1. The shared NPAS server decides where to place global monitoring points and collects link measurements from monitoring points.

2. The shared NPAS server distributes link/path measurements to the NPAS local request server.

3. The NPAS local request server handles applications' requests.

4. The NPAS local request server sends aggregated path request statistics to the shared NPAS server.

Each of above steps consumes a certain amount of network bandwidth. First, we look at the network overhead introduced in step 1. In step 1, the shared NPAS server would need to collect link measurements from the top 5000 shared links' monitoring points. Since the monitoring system only needs to monitor at most 5000 links globally, the monitoring system can collect link measurements at the rate that the monitoring point supports. Even with a 100 times per second collection rate, collecting link measurements of 5000 links consumes only a small amount of bandwidth. For example, if a packet of 100 bytes (800 bits) is used to store the information (e.g., ID, available bandwidth, capacity, loss rate, etc.) of a link and each link information is sent to the shared NPAS server 100 times per second, the total amount of bandwidth that is used network-wide to get information from 5000 shared links is around 400 Mbps.

In step 2, link measurements are distributed to the NPAS local request servers. Since the expected usage pattern of NPAS is that applications will request path advice per flow rather than asking path advice for every packet, the request rate of link measurements by traffic flows is an important factor to determine the distribution rate of link measurements. For example, if a link measurement is requested 100 times every second, the link measurement needs to be updated 100 times per second to provide applications with "up-to-date" link information.

Ideally, the shared NPAS server distributes the link measurements to the local NPAS request server at the rate link information is being requested. As a result, heavily used links get frequent updates while "rarely" used links get less frequent updates. Based on the usage patterns, the measurement on the same link can be distributed at a distinct rate to various ASes.

Table 6.11: The Link Measurement Request and Distribution Rate for the Top 5000 Links

Category	Num Links	Req Rate(times/s)	Dis Rate(times/s)
1	25	>100	100
2	25	>50	50
3	150	>10	10
4	1300	>1	1
5	1000	>0.1	0.1
6	500	>0.033	0.033
7	2000	>0.01	0.01

Based on requests extracted from the UKY trace data, if an application's request contains only a single path, only 5 links are requested more than 100 times per second and thereby only 5 out of 5000 links need to be distributed at the rate of 100 times per second. If all NPAS requests ask for path advice for 5 routing paths, the request rate (Req Rate) and the corresponding link measurement distribution rate (Dis Rate) are described in Table 6.11. We can see that only a small number of links (25 links) are requested by applications more than 100 times per second. Most links are requested

less than 10 times per second, and the information of these links can be distributed at a low distribution rate.

Although each AS gets its own link measurement distribution rates based on the request rates, only a small number of links are requested at a fast rate (e.g., 100 times per second). This is because most traffic is targeted at a few popular destinations. As a result, the link measurement distribution rate for other ASes will likely have a similar distribution pattern, in which measurements of only a few links need to be updated at a fast rate (e.g., 10 times per second to 100 times per second) while measurements of most links only need to be updated at a "normal" rate (e.g., 1 times per second or less than 1 times per second). As a result, link measurements of frequently used links can be updated frequently to help NPAS local request server give correct path advice, and infrequently used links can be updated less frequently to reduce network overhead. In addition, since a type of link metric may be requested at a different rate than other types of link metrics, each type of link metric can have its own link measurement distribution rate. Suppose bandwidth and delay are two of the most important link metrics that all flows requests, and a packet of 8 bytes (80 bits) is used to store the link index (out of 5000 links), the available bandwidth, and the delay of the link. Based on the distribution rate described in Table 6.11, the bandwidth consumption to distribute these two types of link metrics to the NPAS local request server for UKY AS is around 0.5 Mbps. If all other source ASes have request rates similar to the UKY AS, the total bandwidth consumption is around 20Gbps (globally) for 40000 ASes, and the total packet distribution rate is around 4 million per second. Since there are multiple (e.g., dozens of) NPAS shared servers, the global 20Gbps bandwidth consumption can be amortized among all NPAS shared servers to keep the bandwidth consumption at an acceptable level for a NPAS shared server.

In step 3, the application's requests are handled locally by the NPAS local request

server. The NPAS local request server makes use of link measurements distributed by shared NPAS server and information about local links that are collected by the NPAS local request server to give path advice to applications. As a result, requests that have been handled locally do not introduce additional global traffic.

In step 4, the NPAS local request server sends path request statistics to the shared NPAS server. As mentioned earlier, the popular ASes does not change frequently, and so NPAS does not need to update the monitoring points frequently. Assuming that NPAS re-selects the monitoring points every 30 minutes, the NPAS local request server needs to send path request statistics to the shared NPAS server every 30 minutes. The path request statistics of a path include information such as the path ID that is represented as a list of link IDs, a list of metrics that is requested on the path, and the number of requests for each metric. If 100 bytes (800 bits) is used to store the requested statistics of a path and the average number of possible paths between a source and destination is five, the total bandwidth consumption to collect request statistics for 40000 ASes is around 3.3 Gbps (40000 ASes * (39999 * 5) Paths * 800 bits * 1/30 per minute) or 0.32 million packets per second with a packet of size around 1400 bytes.

In conclude, step 3 does not introduce additional traffic globally, and according to step 1, 2, and 4, the shared NPAS server consumes around 3.7 Gbps downstream bandwidth and around 20 Gbps upstream bandwidth, and needs to process around 5 millions packets per second in order to collect and distribute measurement data. Since a single modern server can easily handle 5 million packets per second of IO and maintain concurrent socket connections to 40000 ASes, a dozen of shared NPAS servers distributed across the Internet are sufficient to serve as the core NPAS service and also help distribute the network overhead by sending measurement data to NPAS local request servers from multiple locations (e.g., around 2 Gbps bandwidth consumption per shared NPAS server on average with 12 shared NPAS servers).

In general, the network overhead of providing the NPAS service is acceptable. Deploying dozens of shared NPAS servers can help provide a core NPAS service for a network that is about size of current Internet.

Chapter 7

Collecting Path Information

In the NPAS system, the path requests are collected and aggregated at the NPAS local request server, and then forwarded to the shared NPAS server. To reduce measurement overhead, NPAS only monitors some of the paths; paths that are of interest to the applications. On the other hand, we have shown that NPAS can monitor a small number of links and nodes while still providing link/path information for a high percentage of real network traffic. However, the paths that are of interest to applications change over time. Consequently, NPAS needs to decide how it can update the set of links and ASes that should be monitored. After the monitoring points are selected, NPAS should then determine how frequently the link/path information is collected and distributed. In addition to collecting path information from ISPs (ASes), NPAS also collects information about paths through the NPAS feedback API. So NPAS should know how to handle feedback.

7.1 Changing the Monitoring Points

We have shown that the set of top ASes changes slowly. Consequently, NPAS can use the past several hours' requests to identify the set of links that need to be monitored. However, NPAS also needs to identify an additional set of links to be monitored based on the most recent path requests. ASes that become popular in the past few hours may not otherwise be considered as the top requested ASes. Monitoring links

to those recent popular ASes can help answer requests for ASes that gain temporary popularity or start to become popular.

To include ASes that have recently become popular, NPAS needs to change the set of links/nodes it monitors periodically (e.g., every 30 minutes or every hour). As a result, NPAS periodically (we will use a period of one hour in our example) picks the set of links (SL) to be monitored as follows:

1. Find the top N ASes [1] based on the number of requests in the past several hours (e.g., 12 hours), and place these top ASes in the AS set $A1$.

2. Find all links on the paths from any source AS to ASes in set $A1$, and place these links in link set $L1$.

3. Find M shared links [2] from the link set $L1$ based on the method described in Section 6.3.2, and place these M links into the link set SL.

4. Find the top N ASes based on the number of requests in the past one hour, and place these top ASes that are not in AS set $A1$ in the AS set $A2$.

5. Find all links on the paths from any source AS to ASes in set $A2$, and place these links in the link set $SL2$.

6. Order links in set $SL2$ based on the method as described in Section 6.3.2.

7. Add the first Y new links [3] from the set $SL2$ to SL or until no more new links can be added.

After the popular links are found, NPAS needs to determine the target ASes to be monitored in order to cover these popular links. Based on our previous study in

[1] We set $N = 500$ because more than 80% flows are destined to the top 500 ASes in the UKY trace data.

[2] We set $M = 5000$ based on the experimental results from chapter 6.

[3] We set $Y = 500$ because the popular ASes change slowly and NPAS only needs to monitor a small number of extra links in order to monitor paths to a few ASes that became popular in the past hour.

Chapter 6, NPAS should monitor 50 to 100 ASes if NPAS picks one AS per link or 600 ASes if both sides of a link need to be monitored.

Let CM_{AS} stand for the current set of ASes that are being monitored, and CLh stand for the average link hit ratio since the last update. On each update, given the shared link sets SL, NPAS determines ASes to be monitored as follows:

1. If CLh is less than *MinHitRatio* (where MinHitRatio is the minimum link hit ratio allowed before we regenerate the set of monitored ASes. For example, we can set MinHitRatio=75% to indicate that NPAS wants to provide a minimum link hit ratio of 75%), regenerate the set of ASes to be monitored: clear the AS set CM_{AS}, find MP_M target nodes (or MP_N target nodes if NPAS monitors both endpoints of a link) [4] to be monitored based on the method described in Section 6.4, and put these nodes in CM_{AS}.

2. Put Links that are in SL and can be monitored by monitoring ASes in CM_{AS} in the link set SM. If the number of links in SM is less than M (or M+Y if a maximum of Y recently requested links are considered) [5], additional candidate links are added into SM in sorted order (using the algorithm in Section 6.3.2) until the size of SM reaches M (or M+Y) or no more links can be added (note that we only pick links that can be monitored by monitoring ASes in CM_{AS}).

Although the set of shared links can be updated every 30 minutes (or every hour), there is no need to update target ASes on every update, since enabling a monitoring point in (or collecting information from) a new AS may be expensive. A high link hit ratio usually means NPAS can provide (partial) path information for a high percentage of traffic. Therefore, as described in Step 1, as long as monitoring the current set of ASes can help NPAS achieve a high link hit ratio (e.g., 75% link hit ratio), NPAS does not need to change the set of ASes to be monitored.

[4] we set MP_M=100 and MP_N=600 based on experiment results in Section 6.4.

[5] We set M=5000 and Y = 500 as described in the algorithm to update links to monitor.

7.2 Gathering and Distributing Dynamic Path Measurements

NPAS wants to provide applications with up-to-date path/link information. However, increasing the monitoring rate also means more processing overhead and more network overhead. As described in Section 6.5, NPAS can monitor 5000 shared links at a rate of 100 times per second with a relatively small amount of network bandwidth consumption. Unfortunately, distributing information about 5000 links frequently to all NPAS local request servers may consume a huge amount of network bandwidth. To reduce the distribution traffic overhead, the shared NPAS server can distribute link measurements at a rate based on the link's request rate at the NPAS local request server (as described in Section 6.5). Alternatively, NPAS can use infrequently used paths to retrieve and distribute measurement data. Because NPAS operates in an environment where applications can select paths, NPAS itself can choose paths to distribute the updates that interfere the least with data-plane traffic.

7.3 Getting Path Measurements From Feedback

User feedback contains end-to-end path measurements that were collected by applications and sent to NPAS. Feedback can also indicate the paths that were ultimately used by applications (Note that NPAS gives advice, and, without the feedback service, would not know which pieces of advice the application valued and ultimately used). Using feedback has several benefits. First, NPAS may be able to get end-to-end path information that can not be collected from direct measurements. Second, NPAS learns from feedback about the application's choice of routing path, which gives NPAS an indication of whether applications are satisfied with NPAS's advice, or it helps NPAS recommend the same path to another application with similar requirements.

7.3.1 Handling Feedback

Based on our study of an existing traffic trace from the University of Kentucky, there could be 2000 path requests per second for a single AS. If each request was followed by feedback, there would be 70 million feedback messages per second from all ASes combined. However, not all applications want to provide feedback, nor does NPAS need feedback at such a fast rate. NPAS knows how frequently it needs to update information about a path and uses the feedback API to collect feedback about the path.

Each time NPAS tries to calculate path information from the feedback, it may need multiple feedback updates in order to accurately estimate the path information. To protect user privacy, NPAS collects feedback through the NPAS local request servers. The NPAS local request server can remove the sender's ID in the feedback. The feedback service informs the local request server of the frequency it wants to update path information for a specific path. The NPAS local request server will then try to collect feedback for that path at that frequency. The NPAS local request server is also responsible for verifying the identities of applications that provide feedback.

There are two different types of feedback as described in Section 5.3. First, the feedback can include the actual metric value of a path, such as the the average latency of a path and the maximum sending rate of a path. Second, the feedback can contain the application's choice of routing paths. For example, an application can inform NPAS that one of the recommended paths has actually been used.

For the first type of feedback, the metric value of a path can be calculated based on the average value of path metrics from recent feedback. For the second type of feedback, NPAS learns what paths are picked most by applications from a list of paths, and recommends these most picked paths to applications with similar requirements if NPAS could not give path advice based on its own path measurements. On the other hand, the second type of feedback also informs NPAS whether applications pick the

paths according to the NPAS's advice. Applications may choose a different path if they are not satisfied with the recommended paths. Therefore, the second type of feedback tells NPAS the quality of path advice. If the most picked path is not the highest ranked path, the NPAS local request server may ask the shared NPAS server to distribute the link measurements on involved links more frequently in order to get more accurate ratings of the related paths.

Chapter 8

Rating Paths

After collecting path information, NPAS uses the path information to rate paths for applications. The path information is cached in the NPAS local request server to enable fast response and reduce the bandwidth consumption between the NPAS local request server and the shared NPAS server. In addition, NPAS rates paths efficiently and takes simultaneous competing requests into account when giving out path advice.

8.1 Storing Path Information

The *NPAS information storage service* is responsible for storing and sharing path measurements between NPAS components. Because the goal of NPAS is to provide up-to-date advice based on the most recent information collected, the information storage service does not need to keep outdated path statistics or requests. Instead, the information storage mainly focuses on the recently collected measurements and feedback. However, the information storage service can keep summarized and average statistics, such as the number of requests asking for a specific (partial) path and the average latency of a path.

As shown in Figure 8.1, shared NPAS servers collect path information (line a) and aggregate path requests for paths (line b) through various NPAS collection servers and NPAS local request servers. The information storage service is also responsible for aggregating path requests from the shared NPAS servers, and making aggregated

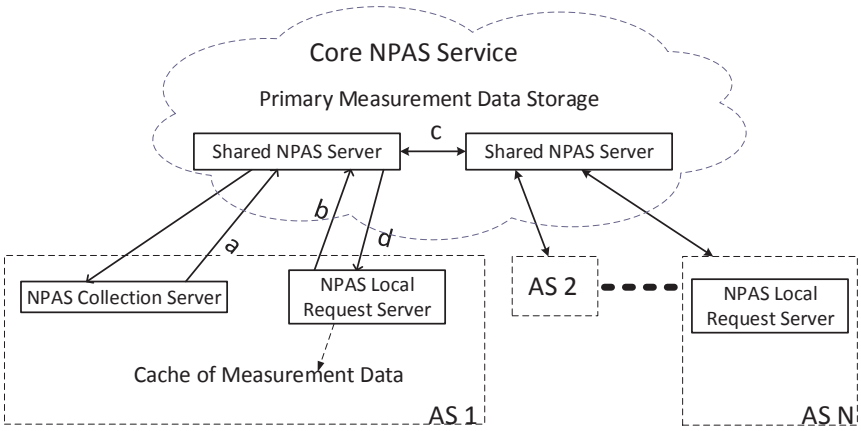

Figure 8.1: The NPAS System Data Flow

information available to the shared NPAS server where the monitoring decision is made (line c). Similarly, the measured path information needs to be shared among the shared NPAS servers in order to be distributed to NPAS local request servers (line c and line d).

The information storage service also works with the NPAS local request server to reduce the bandwidth consumption through caching. In addition, the information storage service does not aggregate the user's private information by using the NPAS local request server as a proxy between applications and the shared NPAS server.

8.1.1 Caching the Link/Path information

The link/path information needs to be delivered to individual NPAS local request servers in order for the advising service to give path advice. As mentioned in Section 6.5, it consumes about 20 Gbps of bandwidth to distribute the information about the top 5000 links. To limit the bandwidth needed to collect and distribute information about infrequently monitored paths, the local request server can cache and reuse the path information for future applications' requests.

To be more specific, for paths that are not covered by frequently measured links. the NPAS local request server determines how long the path information will be

cached and reused. The predicted collection time of the end-to-end path measurement is used as the path measurement expiration time.

Initially, the NPAS local request server only caches the measurements of the frequently monitored shared links, which are distributed by the shared NPAS server at the local request rate. If a request cannot be answered based on the cached link/path information, the NPAS local request server tries to collect feedback about the path information. If the NPAS local request server is able to collect feedback about the requested path, the path information from the feedback will then be cached.

8.1.2 Privacy Issues

Applications make use of NPAS through the NPAS local request server so as to hide their identities from the shared NPAS server. NPAS implementations can have their own ways to protect the user's privacy. For example, in an IP network, an easy way to protect the user's privacy is to replace the source IP address with the IP address of the egress link of the source AS. As a result, the shared NPAS server can still learn what AS-level paths are requested in order to compute what AS-level links need to be monitored, without being able to identify the application's source IP address.

In addition, as described in Section 4.3, by sending the aggregated path requests to the shared NPAS server, the NPAS local request server prevents the shared NPAS server from tracking traffic of individual requests. Furthermore, the local request server can also remove sensitive requests from the aggregated requests or choose to only communicate with trusted core NPAS services to protect privacy.

8.2 Making Path Recommendations

NPAS makes use of the *NPAS advising service* to evaluate paths based on the application's requirements and the path information that NPAS has collected. The NPAS local request server may get several thousands of requests per second.

Consequently, the answer to a request should be calculated in a fast way to keep up with the request rate.

To generate answers to applications that request path advice, the NPAS local request server uses frequently measured link measurements (including the local link measurements) and the cached end-to-end path measurements (e.g., the cached path measurements from feedback) in its local storage to evaluate a path. If only partial path information is available at the time of path request, the NPAS local request server can give the path advice based on partial path information or simply provide the partial path information to applications (note that the NPAS local request server may try to collect feedback about the requested path, and the feedback can be used to provide advice for future requests). In short, the NPAS local request server should provide path advice in an efficient way.

8.3 Handling Competing Requests

Taking recommended routing paths may result in selfish routing [86, 87], and may also lead to traffic oscillations [88, 89] when there are simultaneous requests that share some links. Simultaneous path requests from the same AS and simultaneous path requests from multiple ASes may lead to traffic oscillation if traffic is sent over the "best" path without considering the possibility that the simultaneous requests may congest certain links of the path.

To give intelligent path advice to solve the traffic oscillation problem, the path advice for requests from the same AS are cached at the local NPAS request server. As a result, previous path advice is taken into consideration when new advice is made so that the local NPAS request server can avoid congesting paths. In other words, the second best path may be recommended (i.e., ranked as the best path) when the best path was used by other simultaneous requests. More precisely, the requested bandwidth of the cached advice (assuming the path request had a

bandwidth specification, especially for bandwidth intensive applications) is deducted from the current available bandwidth of a path when rating of the path is calculated for new requests. The cached advice (requests) on a link is cleared when the link measurement is updated (or when the predetermined cache time is reached for infrequently monitored links).

Although the local NPAS request server can cache advice (requests) from the same ASes, simultaneous traffic from other ASes may also congest links that are used by current AS, and thereby lead to traffic oscillation. To solve this problem, the shared NPAS server can set traffic limits on the 5000 globally shared links for ASes. ASes will each get their own limit for the 5000 shared links based on the AS's link usage demand calculated from the path requests. The traffic limit is represented in the form of a percentage value which indicates what percent of available bandwidth an AS can use for simultaneous requests between two updates of the link measurement (e.g., 10 ms). The traffic limit on a link is invoked when the shared NPAS server detects traffic oscillation on that link. The local NPAS request server will suggest the next best path to applications when the limit of some shared link is reached. When the traffic limit on a link is invoked, the local NPAS request server sends the current demand of the link (e.g., how much bandwidth is requested by applications in the most recent 10 milliseconds) to the shared NPAS servers at a fast rate (e.g., 10 to 100 times per second) to help keep the traffic limit for each AS updated. The updated traffic limit on a link is distributed to the local NPAS server along with the link metric update.

8.4 Path Advising Examples

Each NPAS server may have its own algorithm to evaluate paths and give path advice to applications. In the following, we describe a path advising example. For simplicity, we only consider two network metrics: bandwidth and latency. To handle

the path rating request, paths that meet the application's requirements are ordered (in a descending order) by available bandwidth (or in an ascending order if ranking by latency). For example, consider a request asking for path advice on a set of paths with a 1Mbps minimum required bandwidth and a 40ms maximum allowed latency. The local NPAS server will first look at the requested paths whose path information is available in NPAS, and then find the paths that have more than 1Mbps available bandwidth and have no more than 40ms latency. In addition, NPAS orders the paths first by available bandwidth and secondly by latency (note that NPAS can order paths by latency first if the application specifies the latency requirement before the bandwidth requirement in the request to imply that the application is more concerned about latency). If NPAS cannot find a path whose information is available and that also meets the application's requirements, NPAS may choose to return the default path (e.g., the shortest path). NPAS also informs applications whether the path ratings on certain paths are calculated based on partial path information. Applications can either take the recommended paths or make their own decisions based on (partial) path information.

In addition to evaluating paths for applications, the rating API also enables *receiver* to specify constraints on the incoming paths. The constraints are verified at the NPAS local request server where the constraints were received and then forwarded to the shared NPAS server. The shared NPAS server then distributes constraints to other NPAS local request servers. NPAS takes these constraints into consideration when it makes future path advice.

The NPAS system may design its own algorithm to handle scheduling requests. A simple way of handling scheduling requests is to use the greedy algorithm. For example, to schedule a set of flows on a set of paths, the flows can be scheduled one by one, and each time the flow is assigned a highest ranked path based on the requirements. The path information will be updated whenever a flow is assigned

to a path to indicate that the resource required by the flow is reserved. Although this scheduling algorithm may not produce a optimal solution, it can find a possible scheduling. In addition, the handling of scheduling requests is very efficient due to its simplicity.

Chapter 9

Simulation

To evaluate NPAS, we tested NPAS using a simulated network with simulated network traffic. Our goal was to demonstrate that NPAS can provide correct path advice for most requests by keeping track of up-to-date path information. We also studied how applications' performance can be improved by using NPAS, and how the local request server can help reduce network overhead.

9.1 Designing the Simulation System

To simulate a realistic topology, the current Internet was modeled at the AS level. We used the AS relationship data from CAIDA [84] to build the simulation network.

9.1.1 Simulating Network Traffic

We generated simulated traffic to test NPAS' ability to give good path advice. Generating network traffic that can change over time creates a dynamic network environment that NPAS needs to model in order to give correct path advice.

It is infeasible to simulate the network traffic at the packet level because it would require a huge amount of CPU and memory resources. Therefore, we simulated the network traffic at the flow level. We implemented the model described in [90] to generate network traffic for NPAS, and generated a TCP-like flow for each pair of ASes. The amount of network traffic on each AS-level link changes over time in

the simulated network based on traffic pass rate (i.e., the percentage of sent traffic that actually reaches the destination) on a path and the amount of retransmitted traffic [90].

9.1.2 Simulating the Applications' Requests

To make use of NPAS, applications first send path requests to NPAS, and then use the recommended paths to send packets. According to the rating API, a request contains the possible choices of routing paths and the application's requirements for the path. In the simulation, a path was represented by a list of AS IDs, and the application's requirements included the network metric requirements (e.g., how much bandwidth an application needs) and the expected time of using the path.

To simulate realistic requests, the Netflow trace data from the University of Kentucky was used to drive the generation of requests. For each record in the trace, we generated a request that includes multiple paths to the destination in the record and has the same duration of using the path. To test NPAS under different network loads, we varied the size of a flow.

9.1.3 Simulating NPAS

Based on the simulated requests, the algorithm described in Section 7.1 can be used to dynamically decide what monitoring points need to be turned on/off in NPAS. Another job of NPAS is to collect link/path measurements from the monitoring points. Simulating collection of link/path measurements requires us to calculate link/path information in the simulated network.

NPAS is also in charge of collecting feedback from applications. In our simulation, we assumed NPAS can receive feedback from applications. A maximum collecting rate of feedback messages is set to mimic the resource constraint (e.g, computing resource, network resource, etc.) of NPAS. We changed the maximum feedback collection rate in our simulation runs to test NPAS under various network conditions.

In our simulation, NPAS monitored no more than 5000 shared links (by monitoring 100 ASes) and 0-5 local links for most ASes. NPAS evaluated paths that were completely monitored. The "best" evaluated path was given as advice to applications if the "best" evaluated path met the application's requirements or the "best" evaluated path was better than the default path based on the NPAS's knowledge of path information. Otherwise, the default path was used by applications. The monitoring system also took simultaneous requests into consideration when it gave out path advice to prevent congestion on a path while alternative paths were underused.

9.2 Simulation Results

In our simulation, we tested how applications can improve their performance in terms of throughput and latency by taking the NPAS recommended paths. We also studied the percentage of traffic flows that can receive "helpful" path advice from NPAS. In addition, we tried to show the stability of the NPAS advice, the needs of using the NPAS local request server to reduce network overhead, and the benefits of using feedback.

9.2.1 Throughput Improvement

To demonstrate how applications can benefit from NPAS, we calculated the average throughput of applications from various ASes. By comparing the throughput of applications that used the recommended paths, the default paths, and the paths that are picked based on the complete knowledge of the network, we can show how much improvement applications can achieve with the help of NPAS.

In addition to the network traffic as described in Section 9.1.1, the trace traffic was also generated to represent traffic from applications. The trace traffic took the default paths without the use of NPAS. With NPAS, the trace traffic was sent through the recommended paths. Since simultaneous trace traffic may compete with each other

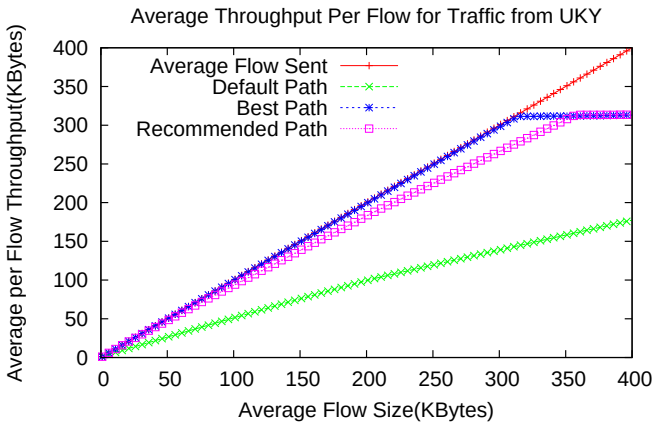

Figure 9.1: Average Throughput Per Flow for Traffic from UKY

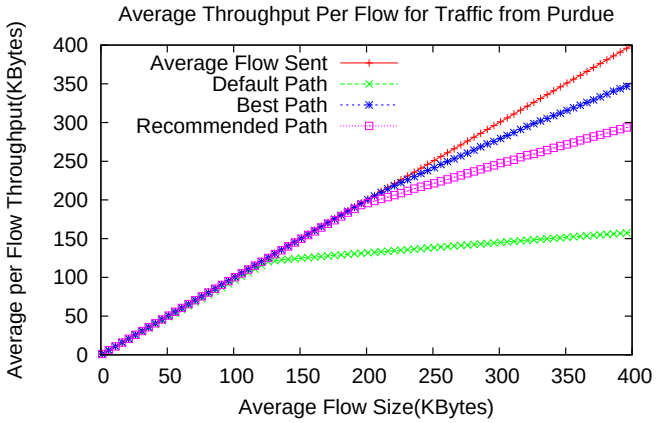

Figure 9.2: Average Throughput Per Flow for Traffic from Purdue

and lead to traffic oscillation, NPAS took the previous requests into consideration when it gives out new path advice as described in Section 8.3.

The actual sending rate of the trace traffic represents the throughput of the applications. In the simulation results, we compared the throughput of applications using distinct paths under various trace traffic loads by changing the amount of traffic per request.

In Figure 9.1 to Figure 9.3, we gradually increased the trace traffic by increasing the flow size from 1Kbytes per flow to 400Kbytes per flow (indicated in the red line), and studied the average per-flow throughput (the actual sending rate) of taking

distinct paths for various source ASes. The trace traffic flows were generated based on our traffic trace data. The green line represents the average throughput per flow if all flows take default paths. The blue line represents the average throughput per flow if all flows can take best paths from the set of possible paths. In other words, the routing decision was made with the knowledge of all the possible paths. The purple line indicates the average throughput if flows are routed based on the NPAS recommended paths.

We can see from Figure 9.1, taking the best path, the average throughput for traffic from UKY is close to the sending rate until the paths are congested. The average throughput of taking the NPAS recommended paths is slightly less than the throughput of taking the best path. The average throughput of taking the default paths is far below the NPAS sending rate. When the per-flow size reaches about 300K bytes, we can see the noticeable congestion on the paths if all traffic is sent via the best path. However, it is not until the per-flow size reaches about 350K bytes, that the paths are congested if traffic is sent using the NPAS recommended path. The average throughput for traffic originating from Purdue (Figure 9.2) has the similar pattern as the throughput for traffic originating from UKY.

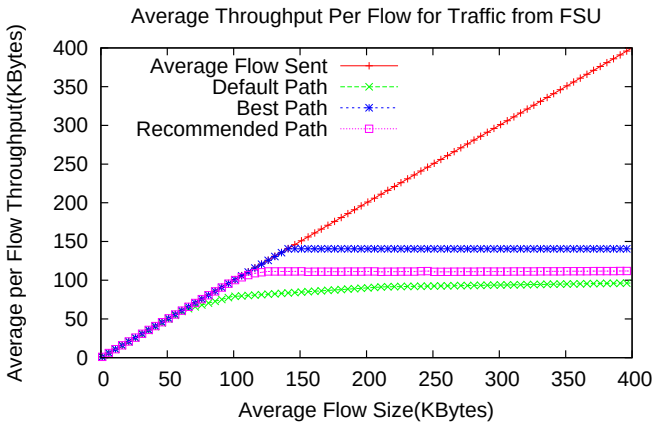

Figure 9.3: Average Throughput Per Flow for Traffic from FSU

In Figure 9.3, the FSU paths become congested with relative small amount

of increment in flow size. But we can still see the benefit of taking the NPAS recommended path before paths are congested.

Similar results can be observed in our test for other representative ASes. In general, making use of NPAS can improve the throughput of applications. The improvement of throughput by taking NPAS recommended path is close to the improvement of throughput by taking the best possible path.

9.2.2 Latency Improvement

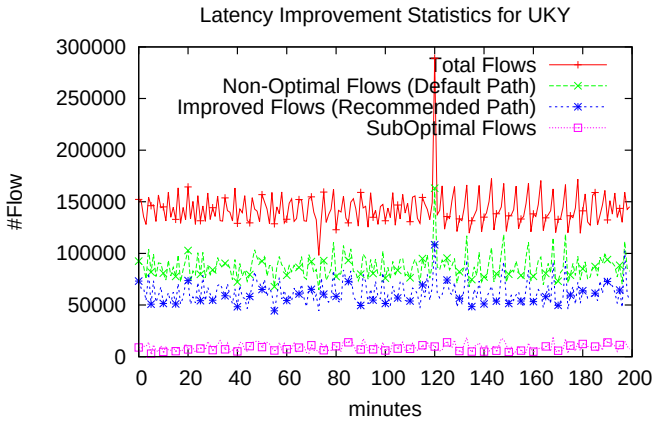

Figure 9.4: Latency Improvement Statistics for UKY

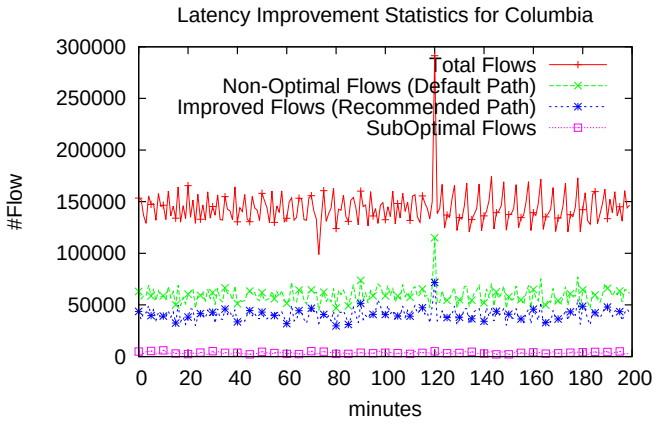

Figure 9.5: Latency Improvement Statistics for Columbia

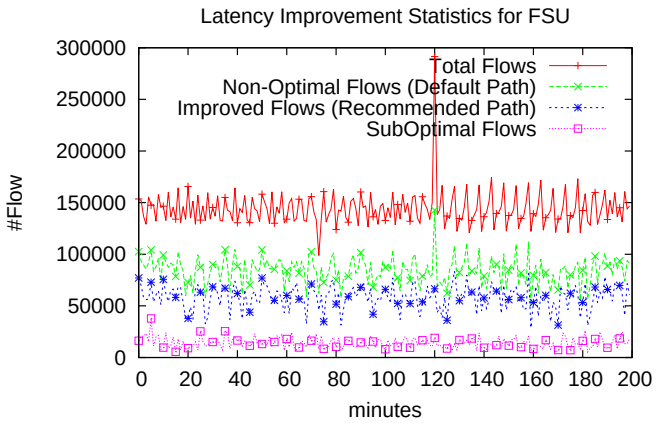

Figure 9.6: Latency Improvement Statistics for FSU

In addition to the throughput, latency is another important network metric that applications need. In our simulated network, we varied link latency over time. For example, at each simulation iteration, for a given link, the link's queuing latency was dynamically picked from a predefined range representing the range of latencies that a flow might experience going over that link. Therefore, we can evaluate how NPAS reduces latency for applications under various network conditions.

In Figure 9.4 to Figure 9.6, we can see the latency improvement statistics over time for these example ASes. The red line indicates the total number of flows. The green lines (*Non-Optimal Flows*) represents the number of flows whose default paths are not the paths with lowest latency. The blue line (*Improved Flows*) represents the number of flows whose latency can be reduced by taking the recommended paths. Although taking the NPAS recommended path can reduce the latency, the latency may still be greater than the latency of the best path. The purple line represents the number of flows that can get lower latency by taking the best path rather than taking the NPAS recommended path. In general, the number of flows that are represented by the purple line is relatively small.

We also tested latency improvement statistics for other university ASes. For most ASes in our test, about 70% of the Non-Optimal flows can get a NPAS recommended

path that has less delay than the default path.

9.2.3 Flow Coverage and Correctness

As shown earlier, applications can improve their throughput by using recommended paths. However, applications may receive distinct path advice for sending each flow. To study the quality of path advice, flows were separated into categories, including helpfully recommended flows (i.e., the recommended path was either the best path or a path that was better than the default path); Non-Optimal flows (i.e., the default path was not the best possible path); improved flows (i.e., flows that got the improved performance by taking the recommended paths rather than the default paths); maximally improved flows (i.e., flows whose recommended paths were not the default paths but were the best possible paths).

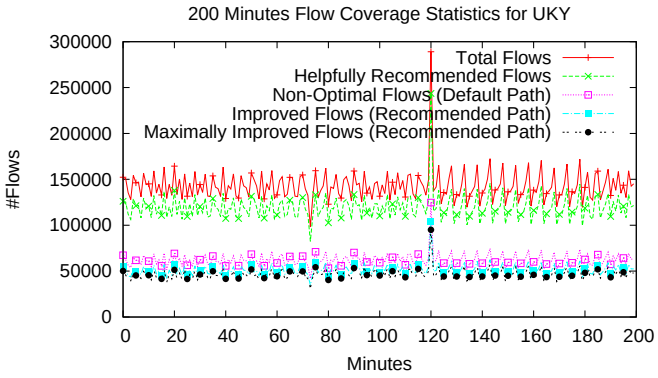

Figure 9.7: 200 Minutes Flow Hit Statistics for UKY

In Figure 9.7 to Figure 9.10, the red, green, purple, light blue, and black line represent the number of total flows, helpfully recommended flows, Non-Optimal flows, improved flows, and maximally improved flows respectively. In these experiments, the achieved bandwidth of a path was used to determine the quality of the path.

We can see from the Figure 9.7, Figure 9.8, and Figure 9.9 that NPAS can provide around 80% of flows with "helpful" advice. In addition, the number of improved flows is also around 80% of the number of Non-Optimal flows, which means

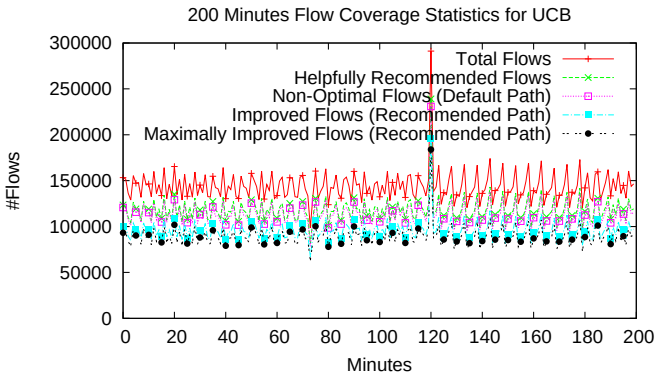

Figure 9.8: 200 Minutes Flow Hit Statistics for UCB

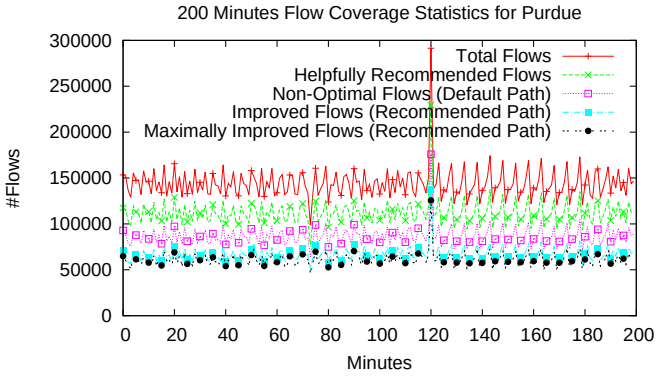

Figure 9.9: 200 Minutes Flow Hit Statistics for Purdue

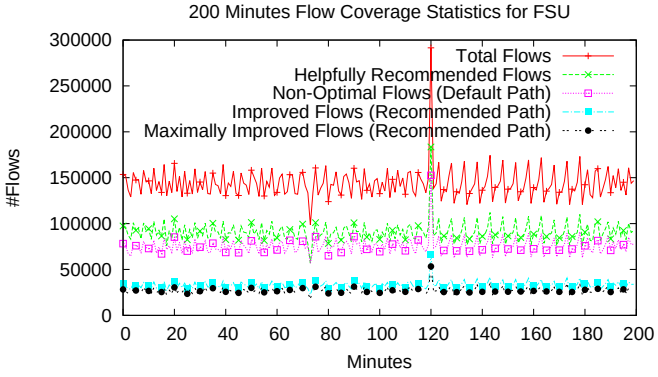

Figure 9.10: 200 Minutes Flow Hit Statistics for FSU

80% of flows that have potential to improve their performance will actually improve their performance by taking the recommended paths. Flows originating from most

representative ASes have a similar pattern to flows originating from these three ASes. We can also see that the number of maximally improved flows is close to the number of improved flows, which implies that if the recommended path is better than the default path, the recommended path is likely to be the best path.

However, a few ASes, such as the FSU AS (shown in Figure 9.10), achieve a lower rate of "helpful" advice. But NPAS is still useful for traffic from these ASes. For example, more than 60% of flows from FSU AS can still get "helpful" advice.

9.2.4 NPAS Stability

It is important for NPAS to provide future applications with stable path advice. Unstable path advice often leads to traffic oscillation. To demonstrate the stability of NPAS's advice, we kept track of the available bandwidth of a set of paths that connect the same pair of ASes when the trace traffic was gradually increased. If NPAS does not give stable advice, the available bandwidth of the chosen set of paths will fluctuate wildly. In contrast, with stable advice, the available bandwidth of the set of paths will gradually come close to each other as the trace traffic increased.

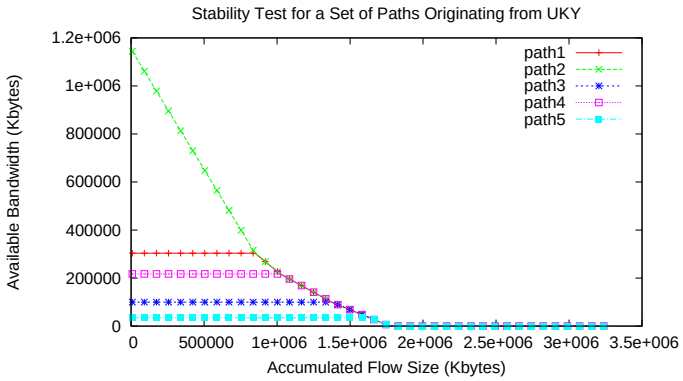

Figure 9.11: Stability Test for a Set of Paths Originating from UKY

Figure 9.11 shows the available bandwidth on a set of paths that are originated from UKY AS and destined to the same destination. We can see that the available bandwidth on these paths come close to each other when the number of flows grows,

and there is no traffic oscillation. Since NPAS takes into account simultaneous requests as described in Section 8.3, it provides stable path advice.

9.2.5 Reduced Network Overhead with Local Request Servers

The NPAS local request server is used to enable fast response and reduce network overhead. Without the NPAS local request server, every request needs to be sent to the shared NPAS server, thereby introducing a great amount of network traffic. With the NPAS local request server, about 60% to 80% requests can be handled with measurements of 5000 globally monitored links. Distributing measurements of 5000 shared links based on request rate only consumes a small amount of bandwidth (described in Section 6.5). In addition, when a path request that cannot be answered based on cached link/path information is received, the NPAS local request server needs to contact the shared NPAS server for the link/path measurements. The NPAS local request server can cache and reuse the link/path information.

Figure 9.12 compares the number of requests that the shared NPAS server needs to handle without the use of a local cache server (a NPAS local request server) to the number of requests the shared NPAS server needs to handle with the use of a local cache server (a NPAS local request server) for the UKY AS. In Figure 9.12, we set cache expiration time to 10 minutes. However, requests for paths that are covered by globally shared links can get much more frequently updated path information (e.g., updated every 10 ms) as described in Section 6.5. Each request that is sent to the shared NPAS server and the returned path measurements introduce additional overhead on the network. On the other hand, if the request can be handled by the NPAS local request server, it keeps the NPAS related traffic local and does not introduce global network overhead.

In Figure 9.12, we can see that the number of requests that need to be sent to the shared NPAS server is very high (indicated in the green line) without the use of the

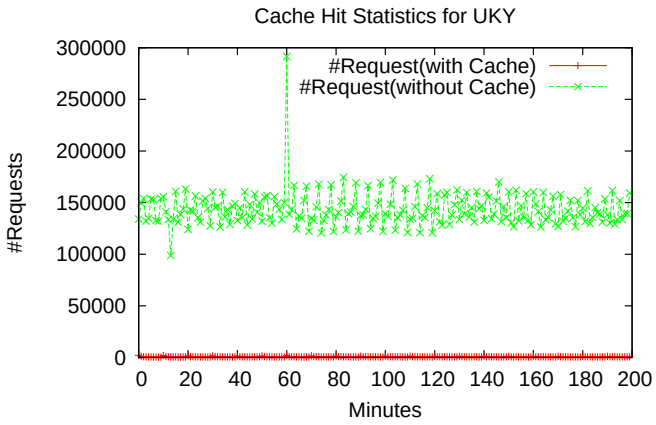

Figure 9.12: Cache Hit Statistics for UKY

NPAS local request server. On the contrary, with the NPAS local request server, the number of requests for which local request server needs to contact the shared NPAS server is relatively low (indicated in the red line in Figure 9.12). In general, the use of the NPAS local request server can greatly help reduce the global network overhead.

9.2.6 Benefits of Using Feedback

Making use of feedback that contains path/link performance measurements can help improve the advice NPAS gives out. On the other hand, collecting feedback also introduces additional network overhead.

Path measurements that are obtained from feedback may be outdated at the time that the path measurements are used to give path advice. However, if the path measurements that are collected from feedback are still accurate when they are used to rank paths, the correct rate of NPAS advice can be improved by collecting and using feedback.

Figure 9.13 shows how the number of feedback messages that is collected by NPAS affects the 3-path hit ratio in a network that supports 5 paths for our representative university ASes in the US. In our simulation, we assumed the feedback can be used to correctly rate a path, and NPAS collected one feedback message per path. In reality,

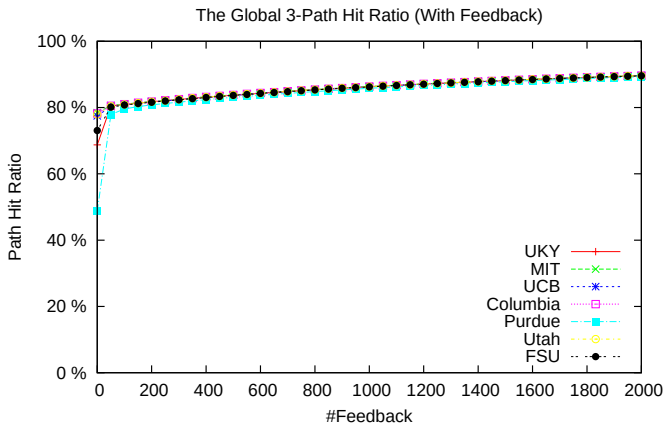

Figure 9.13: 3-Path Hit Ratio With Feedback for US ASes

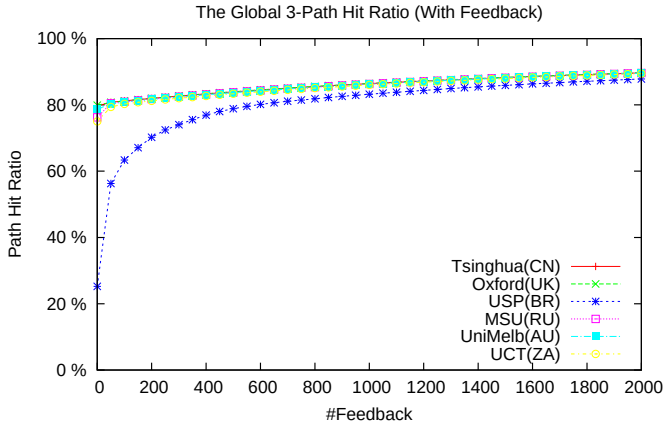

Figure 9.14: 3-Path Hit Ratio With Feedback for International ASes

NPAS may need to collect multiple feedback messages for the same path in order to accurately estimate the path information. The unmonitored paths were sorted according to how frequently the paths were requested by applications, and NPAS collected feedback on a predetermined number of paths according to their sorted order.

As we can see from Figure 9.13, collecting 2000 feedback messages per AS can help most ASes achieve a 90% 3-path hit ratio. Comparing to the path hit ratio without the use of feedback, collecting 2000 feedback messages provides around 10% to 40% improvement for 3-path hit ratio. Figure 9.14 shows the 3-path hit ratio with

feedback for International ASes. With 2000 feedback messages, most ASes reach around a 3-path hit ratio of 90%.

We can see that feedback provides a reasonable improvement on path hit ratio. However, collecting feedback may require the cooperation of applications and consume additional network resources. Therefore, ASes that already achieve high path hit ratios may decide not to collect feedback. In other words, NPAS can decide whether it needs to collect feedback and how many unmonitored paths it would like to collect feedback for.

Chapter 10

Conclusions

Future networks will likely provide applications with the ability to select the path(s) their data traverses between a source and destination. Because of the huge overhead of frequently monitoring all paths, future routing services will not be able to rank paths for applications. To help future applications select among potential routing paths, we proposed a new *network path advising service (NPAS)*. NPAS is designed as a supplementary service to the routing service. Applications use the routing service to get possible paths between two end hosts, and then use NPAS to rate paths based on path information and the applications' requirements (e.g., QoS requirements). Based on the rating results, an application can pick routing paths that have high ratings and meet the application's requirements.

Comparing to the traditional monitoring systems, NPAS has several advantages. First, NPAS is designed specifically for applications rather than for network administrators. Second, the paths being monitored are determined based on applications' requests. Third, application feedback is used to improve path evaluation. Fourth, the receiver can tell NPAS its preferences about incoming paths. Fifth, NPAS can scale to the size of the Internet while still giving correct advice for around 75% of traffic.

NPAS leverages a distributed architecture to deliver shared path/link information and local path/link information to applications. The application's requests will be handled locally to enable fast response and help NPAS scale. Only the aggregated

statistics of requests need to be sent to shared NPAS servers to help NPAS decide monitoring points. The information of the shared paths/links is distributed by shared NPAS servers to NPAS local request servers at a rate based on the request frequency. The NPAS local request server can evaluate paths for applications based on the shared path information and local path information. In addition, NPAS provides a set of APIs to help applications rank/rate paths, schedule traffic across multiple paths, and report feedback.

Based on the analysis of the real-world traffic trace, NPAS is able to develop algorithms to identify a small number of shared links (i.e., 5000 links) and nodes (i.e., 100 to 600 nodes) to monitor while still "covering" a large percent traffic (e.g., 75% of traffic) with the help of a local monitoring service. The link/path information needs to be collected frequently in order to provide applications with up-to-date information. Since NPAS only monitors a small number of shared links (i.e., 5000 links), it can monitor these links frequently (e.g., 100 times per second) without overloading the network. Information about shared links/paths alone may not cover the entire path, because key local links for an AS may not be used by other ASes and thereby are not monitored globally. As a result, NPAS also monitors local links in order to provide an end-to-end view of a path to applications. The local link information is kept at the NPAS local request server and is not shared among all ASes.

Using the simulated network, we demonstrated that NPAS will help applications improve their throughput and latency. In addition, NPAS provided around 80% of flows with "helpful" advice for most ASes we evaluated.

10.1 Future Work

NPAS shows great potential to provide beneficial path advice to future applications, but there remains aspects of NPAS that need additional study. The following outlines future work that would help produce a more complete NPAS:

Transitioning from the current Internet

In current IP networks, the source routing approach is often disabled. To provide applications with choices of routing paths, a modern network architecture, such as ChoiceNet [77], will need to be deployed. Although it is difficult to build a complete new Internet architecture, we can always deploy source routed networks using overlay networks. The applications that need to pick a custom path can join the overlay network.

NPAS is designed for applications that are running on a network the size of the current Internet. We mentioned earlier that NPAS can pay ISPs to collect path information. On the other hand, ISPs may be willing to provide path information in order to attract traffic and make money by carrying traffic though their domains. The commercial model needs to be further studied before NPAS can be deployed. For example, the responsibility of each party and the payment mechanism need to be addressed. In addition, the protocol for NPAS to collect path information from ISPs also needs to be defined.

To help applications use NPAS, a library will need to be developed. There could be two types of libraries: the library that applications can invoke directly to use the NPAS service, and the library that is used by the operating system to support transparent use of the NPAS service for applications.

Giving Intelligent Path Advice

This thesis focuses on determining what nodes/links to monitor and on how to collect path/link information, rather than making intelligent use of path/link information when giving path advice. Although NPAS may not have complete path information, making intelligent use of partial path information may be able to identify whether one path is better than another. For example, if two paths share some links and all links on the paths except the shared links are

monitored, NPAS would be able to compare these two paths. Alternatively, if NPAS knows that the link that NPAS does not have information about is not a bottleneck link (through some out-of-band methods or learning from the historical data), NPAS may still be able to evaluate paths that containing the link without knowing the current information of that link.

Another possibility for giving intelligent path advice is to predict the future path information based on the current path information, path requests and historical path statistics. The prediction about path information can be used to improve the advice NPAS gives.

The NPAS APIs also enable receivers to specify constraints on the incoming paths. Since NPAS is designed for Internet-scale systems, we need to study what constraints are allowed, how constraints are updated in NPAS, and how to use these constraints to give path advice.

Providing Complete Path Information

NPAS may also need to collect links that are requested infrequently in order to provide complete path information for applications. An algorithm needs to be designed to determine the collection frequency of these links. In addition, how information of these infrequently requested links is distributed in the NPAS architecture needs to be addressed.

Collecting and Verifying Feedback

Feedback can help NPAS learn about a path. However, feedback can also be unwanted, outdated or inaccurate. To make better use of feedback, we need to study how NPAS can filter out "wrong" (e.g., unwanted, outdated, inaccurate) feedback. NPAS may need to collect a certain amount of feedback messages from a variety of applications on a path in order to estimate the path

information. Therefore, NPAS may need to decide what applications it wants to collect feedback from.

Security

Security is not addressed in this thesis, but security mechanisms need to be designed to ensure a safe and reliable future network path advising service. NPAS needs to be able to detect malicious applications. For example, applications may send fake requests to fool NPAS into monitoring some unwanted links. In addition, applications may flood the local NPAS servers with requests. Consequently, we must design a security model to help NPAS identify applications with unusual usage patterns, and prevent these applications from using NPAS.

Bibliography

[1] Border Gateway Protocol. http://en.wikipedia.org/wiki/Border_Gateway_Protocol.

[2] Open Shortest Path First. http://en.wikipedia.org/wiki/Open_Shortest_Path_First.

[3] The Global Environment for Network Innovations. http://www.geni.net.

[4] T. Anderson, K. Birman, R. Broberg, M. Caesar, D. Comer, C. Cotton, M. Freedma, A. Haeberlen, Z. Ives, A. Krishnamurthy, and et al. NEBULA - A Future Internet That Supports Trustworthy Cloud Computing. http://nebula-fia.org/papers/NEBULA-WP_TOC.pdf.

[5] A. Seehra, J. Naous, M. Walfish, D. Mazieres, A. Nicolosi, and S. Shenker. A Policy Framework for The Future Internet. In *ACM Hotnets*, 2009.

[6] A. Anand, F. Dogar, D. Han, B. Li, H. Lim, M. Machado, W. Wu, A. Akella, D.G. Andersen, J.W. Byers, S. Seshan, and P. Steenkiste. XIA: An Architecture for an Evolvable and Trustworthy Internet. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. ACM, 2011.

[7] K.L. Calvert, J. Griffioen, and L. Poutievski. Separating Routing and Forwarding: A Clean-Slate Network Layer Design. In *Broadband Communications, Networks and Systems, 2007. BROADNETS 2007. Fourth International Conference on*. IEEE, 2007.

[8] Open Networking Fundation. Software-defined Networking: The New Norm for Networks. *ONF White Paper*, 2012.

[9] O. Ascigil, K.L. Calvert, and J. Griffioen. On the Scalability of Interdomain Path Computations. In *Proceedings of the 13th IEEE/IFIP Networking*, 2014.

[10] Network Monitoring. http://en.wikipedia.org/wiki/Network_monitoring.

[11] A. Hanemann, J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. Łapacz, D. Swany, S. Trocha, and J. Zurawski. Perfsonar: A Service Oriented Architecture for Multi-domain Network Monitoring. *Service-Oriented Computing-ICSOC 2005*, pages 241–254, 2005.

[12] Archipelago Measurement Infrastructure. http://www.caida.org/projects/ark.

[13] V. Jacobson. Pathchar: A Tool to Infer Characteristics of Internet Paths, 1997.

[14] M. Jain and C. Dovrolis. Pathload: A measurement Tool for End-to-End Available Bandwidth. In *In Proceedings of Passive and Active Measurements (PAM) Workshop*, 2002.

[15] C. Dovrolis, P. Ramanathan, and D. Moore. What do Packet Dispersion Techniques Measure? In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications. IEEE*, 2001.

[16] Flow-tools. http://linux.die.net/man/1/flow-tools.

[17] K. Lai and M. Baker. Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In *USITS*, 2001.

[18] S. Seshan, M. Stemm, and R.H Katz. SPAND: Shared passive network performance discovery. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

[19] A. Coates, A.O. Hero III, R. Nowak, and B. Yu. Internet Tomography. *Signal Processing Magazine, IEEE*, 19(3):47–65, 2002.

[20] Y. Chen, D. Bindel, and R.H. Katz. Tomography-Based Overlay Network Monitoring. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*. ACM, 2003.

[21] S. Chen and K. Nahrsted. An Overview of Quality of Service Routing for Next-generation High-speed Networks: Problems and Solutions. *Special Issue on Transmission and Distribution of Digital Video, IEEE Network*, 12(6):64–79, 1998.

[22] Z. Wang and J. Crowcroft. Quality-of-service Routing for Supporting Multimedia Applications. *Selected Areas in Communications, IEEE Journal on*, 14(7):1228–1234, 1996.

[23] Q. Ma and P. Steenkiste. Quality-of-service Routing for Traffic with Performance Guarantees. In *Building QoS into Distributed Systems*, pages 115–126. 1997.

[24] C.R. Lin and J. Liu. QoS Routing in ad hoc Wireless Networks. *Selected Areas in Communications, IEEE Journal on*, 17(8):1426–1438, 1999.

[25] A. Nakao, L. Peterson, and A. Bavier. A Routing Underlay for Overlay Networks. In *Proceedings of ACM SIGCOMM*. ACM, 2003.

[26] T. Fei, S. Tao, L. Gao, and R. Guerin. How to Select a Good Alternate Path in Large Peer-to-Peer Systems? In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. IEEE*, 2006.

[27] P.K. Gummadi, H.V. Madhyastha, S.D. Gribble, H.M. Levy, and D. Wetherall. Improving the Reliability of Internet Paths with One-hop Source Routing. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation*, 2004.

[28] D. Harrington, R. Presuhn, and B. Wijnen. RFC 3411: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. *RFC*, 2002.

[29] NetFlow. http://en.wikipedia.org/wiki/NetFlow.

[30] T. Wolf, R. Ramaswamy, S. Bunga, and Ning Yang. An Architecture for Distributed Real-Time Passive Network Measurement. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on*, 2006.

[31] P. Francis, S. Jamin, V. Paxson, L. Zhang, D.F. Gryniewicz, and Y. Jin. An Architecture for A Global Internet Host Distance Estimation Service. In *INFOCOM 1999. Eighteenth Annual Joint Conference of the IEEE Computer and Communications. IEEE*, 1999.

[32] T.S.E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2002.

[33] Content Distribution Network. http://en.wikipedia.org/wiki/Content_delivery_network.

[34] A.B. Downey. Using Pathchar to Estimate Internet Link Characteristics. *ACM SIGCOMM Computer Communication Review*, 29(4):241–250, 1999.

[35] K. Lai and M. Baker. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay. *ACM SIGCOMM Computer Communication Review*, 30(4):283–294, 2000.

[36] M. Jain and C. Dovrolis. Pathload: A Measurement Tool for End-to-End Available Bandwidth. In *Proceedings of Passive and Active Measurements (PAM) Workshop*, 2002.

[37] J. Strauss, D. Katabi, and F. Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet Measurement*. ACM, 2003.

[38] K. Harfoush, A. Bestavros, and J. Byers. Measuring Bottleneck Bandwidth of Targeted Path Segments. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE*, 2003.

[39] R.L. Carter and M.E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. *Performance evaluation*, 1996.

[40] V. Paxson. End-to-end Internet Packet Dynamics. In *ACM SIGCOMM Computer Communication Review*, volume 27, pages 139–152. ACM, 1997.

[41] K. Lai and M. Baker. Measuring Bandwidth. In *INFOCOM 1999. Eighteenth Annual Joint Conference of the IEEE Computer and Communications. IEEE.* IEEE, 1999.

[42] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *Passive and Active Measurement Workshop*, 2003.

[43] N. Hu and P. Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. *Selected Areas in Communications, IEEE Journal on*, 21(6):879–894, 2003.

[44] B. Melander, M. Bjorkman, and P. Gunningberg. A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks. In *Global Telecommunications Conference, 2000. GLOBECOM'00. IEEE*. IEEE, 2002.

[45] A. Gerber, J. Pang, O. Spatscheck, and S. Venkataraman. Speed Testing Without Speed Tests: Estimating Achievable Download Speed from Passive Measurements. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2010.

[46] V. Sekar, M.K. Reiter, and H. Zhang. Revisiting the Case for a Minimalist Approach for Network Flow Monitoring. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2010.

[47] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S.C. Diot. Packet-Level Traffic Measurements from The Sprint IP Backbone. *Network, IEEE*, 17(6):6 – 16, 2003.

[48] Domain Name System. http://en.wikipedia.org/wiki/Domain_Name_System.

[49] K.P. Gummadi, S. Saroiu, and S.D. Gribble. King: Estimating Latency Between Arbitrary Internet End Hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*. ACM, 2002.

[50] D. Leonard and D. Loguinov. Turbo King: Framework for Large-Scale Internet Delay Measurements. In *INFOCOM 2008. 27th Conference on Computer Communications. IEEE*, 2008.

[51] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[52] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *Selected Areas in Communications, IEEE Journal on*, 22(1):41–53, 2004.

[53] Li Gong. JXTA: A Network Programming Environment. *Internet Computing, IEEE*, 5(3):88–95, 2001.

[54] Extensible Messaging and Presence Protocol. http://en.wikipedia.org/wiki/XMPP.

[55] I. Clarke, O. Sandberg, B. Wiley, and T.W Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies*, 2001.

[56] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-generation Onion Router. Technical report, DTIC Document, 2004.

[57] Gnutella. http://en.wikipedia.org/wiki/Gnutella.

[58] S.J. Lee, S. Banerjee, P. Sharma, P. Yalagandula, and S. Basu. Bandwidth-Aware Routing in Overlay Networks. In *INFOCOM 2008. 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[59] Virtual LAN. http://en.wikipedia.org/wiki/Virtual_LAN.

[60] Virtual Private Network. http://en.wikipedia.org/wiki/Virtual_private_network.

[61] Virtual Private LAN Service. http://en.wikipedia.org/wiki/Virtual_Private_LAN_Service.

[62] W.D. Laverell, Z. Fei, and J. Griffioen. Isn't It Time You Had an Emulab? *ACM SIGCSE Bulletin*, 40(1):246–250, 2008.

[63] PlanetLab. https://www.planet-lab.org.

[64] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In VINI Veritas: Realistic and Controlled Network Experimentation. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 3–14. ACM, 2006.

[65] A. Ciuffoletti. Monitoring a Virtual Network Infrastructure. *ACM SIGCOMM Computer Communication Review*, 2010.

[66] J. Sommers and P. Barford. An Active Measurement System for Shared Environments. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2007.

[67] G. Liang and B. Yu. Maximum Pseudo Likelihood Estimation in Network Tomography. *Signal Processing, IEEE Transactions on*, 51(8):2043–2053, 2003.

[68] Y. Tsang, M. Coates, and R.D. Nowak. Network Delay Tomography. *Signal Processing, IEEE Transactions on*, 51(8):2125–2136, 2003.

[69] Y. Tsang, M. Coates, and R. Nowak. Passive Network Tomography Using EM Algorithms. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. IEEE, 2001.

[70] A. Chen, J. Cao, and T. Bu. Network Tomography: Identifiability and Fourier Domain Estimation. In *INFOCOM 2007. Twenty-Sixth Annual Joint Conference of the IEEE Computer and Communications. IEEE*, 2007.

[71] N.G Duffield and F.L. Presti. Network Tomography from Measured End-to-End Delay Covariance. *IEEE/ACM Transactions on Networking (TON)*, 12(6):978–992, 2004.

[72] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network tomography: recent developments. *Statistical science*, pages 499–517, 2004.

[73] J.D. Horton and A. López-Ortiz. On the Number of Distributed Measurement Points for Network Tomography. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*. ACM, 2003.

[74] R. Kumar and J. Kaur. Efficient Beacon Placement for Network Tomography. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2004.

[75] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. RFC 2205: Resource ReSerVation Protocol (RSVP). *RFC*, 1997.

[76] M. Yang, Y. Huang, J. Kim, M. Lee, T. Suda, and M. Daisuke. An End-to-End QoS Framework with On-Demand Bandwidth Reconfiguration. *Computer Communications*, 28(18):2034–2046, 2005.

[77] T. Wolf, J. Griffioen, K.L. Calvert, R. Dutta, G.N. Rouskas, I. Baldine, and A. Nagurney. Choice as a Principle in Network Architecture. *ACM SIGCOMM Computer Communication Review*, 42(4):105–106, 2012.

[78] S. Bhardwaj, L. Jain, and S. Jain. Cloud Computing: A Study of Infrastructure as a Service (IAAS). *International Journal of engineering and information Technology*, 2(1):60–63, 2010.

[79] Platform as a Service. http://en.wikipedia.org/wiki/Platform_as_a_service.

[80] Mehmet Onur Ascigil. *Design of a Scalable Path Service for the Internet*. PhD thesis, University of Kentucky, May 2015.

[81] Facebook. http://en.wikipedia.org/wiki/Facebook.

[82] NetFlix. http://en.wikipedia.org/wiki/Netflix.

[83] Youtube. http://en.wikipedia.org/wiki/YouTube.

[84] CAIDA. http://www.caida.org/.

[85] Global Internet Phenomena Report. https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/2h-2013-global-internet-phenomena-report.pdf.

[86] L. Qiu, Y.R. Yang, Y. Zhang, and S. Shenker. On Selfish Routing in Internet-like Environments. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM, 2003.

[87] T. Roughgarden and É. Tardos. How Bad Is Selfish Routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

[88] R. Keralapura, C. Chuah, N. Taft, and G. Iannaccone. Race Conditions in Coexisting Overlay Networks. *Networking, IEEE/ACM Transactions on*, 16(1):1–14, 2008.

[89] Y. Liu, H. Zhang, W. Gong, and D. Towsley. On the Interaction between Overlay Routing and Underlay Routing. In *INFOCOM 2005. Twenty-Fourth Annual Joint Conference of the IEEE Computer and Communications. IEEE*, 2005.

[90] S. Wei and J. Mirkovic. A Realistic Simulation of Internet-scale Events. In *Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools*. ACM, 2006.

Vita

- Education
  - Xiamen University, Xiamen, Fujian, China, *M.S. in Computer Application Technology*, Sep. 2004 – July 2007
  - Xiamen University, Xiamen, Fujian, China, *B.Eng. in Software Engineering*, Sep. 2000 – July 2004
- Employment History
  - University of Kentucky, Lexington, KY, Research Assistant, May. 2008 - Aug. 2012 & Jan. 2013 – Feb. 2015
  - University of Kentucky, Lexington, KY, Teaching Assistant, Aug. 2007 – May. 2018 & Aug. 2012 – Dec. 2012
- Publications
  - X. Wu, J. Griffioen, "Supporting Application-based Route Selection", *In Proceedings of 23nd International Conference on Computer Communications and Networks (ICCCN), IEEE, 2014*
  - X. Wu, J. Griffioen, "Network path advising service for the future Internet", *In Network Operations and Management Symposium (NOMS), IEEE, 2012*
  - J. Griffioen, Z. Fei, H. Nasir, X. Wu, J. Reed, C. Carpenter, "Measuring Experiments in GENI", *In Computer Networks, 2013*
  - J. Griffioen, Z. Fei, H. Nasir, X Wu, J. Reed, C. Carpenter, "The design of an instrumentation system for federated and virtualized network testbeds", *In Network Operations and Management Symposium (NOMS), IEEE, 2012*
  - J. Griffioen, Z. Fei, H. Nasir, X. Wu, J. Reed, C. Carpenter, "Teaching with the Emerging GENI Network", *In Proceedings of the 2012 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS), 2012*
  - J. Duerig, R. Ricci, L. Stoller, M. Strum, G. Wong, C. Carpenter, Z. Fei, J. Griffioen, H. Nasir, J. Reed and X. Wu, "Getting started with GENI: a user Tutorial", *In ACM SIGCOMM Computer Communication Review, 2012*