Measuring the Internet

Measuring the Internet

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 20 november 2006 om 10.00 uur

door

Xiaoming ZHOU

elektrotechnisch ingenieur
geboren te Lian Pin, Guangdong Province, China.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. P.F.A. Van Mieghem

Samenstelling promotiecommissie:

Rector Magnificus,	Voorzitter
Prof.dr.ir. P.F.A. Van Mieghem,	Technische Universiteit Delft, promotor
Prof.dr.ir. I.G.M.M. Niemegeers,	Technische Universiteit Delft
Prof.dr.ir. N.H.G. Baken,	Technische Universiteit Delft
Prof.dr.ir. W. Vree,	Technische Universiteit Delft
Dr. H.A.J.R. Uijterwaal,	Réseaux IP Européens NCC
Prof.dr.ir P. Demeester,	Ghent University
Prof.dr.ir G. Leduc,	University of Liège

ISBN-13: 978-90-5335-101-7
ISBN-10: 90-5335-101-9

Keywords: Internet measurement, Network layer, Application layer

Printed in The Netherlands

to Ye Zhuyin, Zhou Fuchang, Zhou Xiaohui, and Zou Wenzhu

Contents

Summary

Title: Measuring the Internet

The Internet is a collection of networks that use the TCP/IP suite of protocols. It has a huge impact on human activity. There are currently hundreds of millions of computers connected to the Internet, generating several petabytes traffic a day. Internet is still growing rapidly. However, the Internet today is not yet precisely characterized. One reason for this is that it is dynamic, constantly changing in size, traffic load, and application types. Recently, there has been a lot of effort put into various aspects of Internet measurements. These are important to the scientists as they provides crucial, fundamental knowledge about Internet structure and performance, and, at the same time, these measurements have added value for Internet Service Providers (ISPs) in terms of service monitoring and for management purposes.

Results obtained so far in Internet measurements are very encouraging. Significant progress has been made in many fields (*e.g.*, we now understand the topology of Web much better than before), but there are still many aspects of the Internet's structure, workload, and applications that are unexplored. This thesis addresses several unanswered questions about the performance of the Internet at the network and the application layer. To mention a few:

1. How can we model the Internet infrastructure, and how can this be measured?

2. How does IPv6 compare to IPv4 in terms of delay and loss performance and how has performance evolved over the past few years?

3. How can we evaluate user application performance through Internet measurements?

4. Is there any method to estimate network distance based on reduced or incomplete measurements (*e.g.*, delay and hopcount)?

We intend to address these questions by measuring the Internet and analyzing empirical evidence obtained from Internet data. In this thesis, we show that accurate measurements not only enhance our understanding of the current Internet, but can

also lead to recommendations for improvements on both the network infrastructure and network protocols.

The major contributions of this thesis can be divided into the following three parts:

First, in Chapter 2, we present what is known about the measurement's framework and metrics. We also analyze how measurement's uncertainties and errors influence Internet measurement. By using well defined measurement's framework and metrics, and taking their uncertainties and errors into account, we obtain more accurate large-scale Internet measurements.

Second, we evaluate Internet performance through real Internet measurements at the network layer. There, we focus on a set of standard metrics that can be applied to measure the quality, performance, and reliability of Internet data delivery services. These metrics include connectivity, one-way delay and loss, delay variation, and packet reordering (*i.e.*, the out-of-order arrival of packets at the destination). In Chapter 3, we show that the hopcount distribution in the Internet (the distribution of path lengths in hops) can be modeled by that of a random graph with uniformly or exponentially link weights. We also show that for large group sizes, the node degree distribution apparently obeys a power-law, while for small group sizes, the node degree distribution appears better fitted with exponential distribution. Our experimental results in Chapter 4 show that reordering may significantly impact on the performance of applications in the Internet since reordering increases a high delay cost for recovery on the end host. We also show that reordering depends on the network load. Chapter 5 examines the IPv6 infrastructure by comparing the IPv6 and IPv4 delay and loss evolution measurements under the current network situations. We show that the average loss of IPv6 is only slightly worse than IPv4, while the delay in the case of IPv6 is clearly much worse.

Third, we investigate how the application qualities perceived by the end users (at the application layer) are influenced by the network-layer performance. Applications are the visible part of the Internet for millions of its users. Voice over IP (VoIP) and Peer-to-Peer (P2P) are two applications that have recently attracted a lot of attention due to their wide deployment. The novel aspect in this part is the estimation of the VoIP perceived quality and the estimation of P2P distance through real Internet measurement. In Chapter 6, we show how different codecs can affect the perceived voice quality, and that the current high speed networks can continuously achieve a high VoIP quality. In Chapter 7, we study a method for estimation of network distance (*e.g.* hopcounts or delays) based on reduced or incomplete distance measurements. This is done by assigning coordinates to estimate the network distance between any two hosts. Our large scale measurements demonstrate that the method is accurate and scalable. Applications that can benefit from such knowledge include content delivery networks, P2P networks, and multiuser games.

Author: Xiaoming Zhou

Chapter 1

Introduction

1.1 Why Measuring the Internet

The Internet uses the TCP/IP suite of protocols to interconnect computers with each other. It has an enormous impact of the activities of human being: In 2006, hundreds of millions of computers are connected to the Internet, generating about several petabytes traffic a day. The Internet is still growing rapidly. However, the Internet today has not been precisely measured. One reason is that it is dynamic: it is constantly changing in size, traffic, and application. Internet measurement is a relatively new field but is playing a key-role in providing crucial, fundamental knowledge of the Internet to both researchers and Internet Service Providers (ISP).

Internet measurement is important for Internet researchers. Unlike mathematical and simulation models, Internet measurements provide a "real check" of the Internet properties, and thus provide an in-depth understanding of the Internet performance. Literature survey reveals that Internet measurements are of increasing interest. For example, the Citeseer database[1] [60] compiles a list of the most-cited papers published in each year. Since 1993, Internet measurement papers have been among the top 20 most-cited published each year. Driven by Internet measurement, network research is now being evaluated in a more qualitative and quantitative way.

Statistical analyses of these measurement data have revealed some unexpected structural features of the Internet:

(1) The node degree distribution of the Internet apparently obeys a power law [34][19][97][61][14][52];

(2) There are typically short distances between arbitrary pairs of nodes [106]. It has also been shown that many networks have similar small-world property [13][3];

(3) Routing in practice can often be sub-optimal, in both efficiency and reliability [6][90][92].

[1] CiteSeer database put papers available in digital form on the web.

Furthermore, the landmark work of Vern Paxson [76] discusses findings from a large-scale study of Internet packet dynamics conducted by tracing 20,000 TCP bulk transfers between 35 Internet sites. The measurement results have shown the asymmetric property of the end-to-end behaviors due to the different directions of the Internet paths. The work has investigated the prevalence of unusual network events such as out-of-order delivery (reordering) and packet corruption; and also investigated patterns of packet loss, finding that loss events are not well-modeled as independent and, furthermore, that the distribution of the duration of loss events exhibits infinite variance. Finally, the work has analyzed variations in packet transit delays as indicators of congestion periods, finding that congestion periods also span a wide range of time scales. This result had significant impact on the later network research [77][62][109][54].

Internet measurements are also important for ISPs. For example, service monitoring and management guarantee Service Level Agreements (SLA) in the dynamic Network. The measurements of traffic volumes provide billing data. Furthermore, Internet measurement improves understanding of traffic variability and Network growth, allowing ISPs to improve network performance and the Network planning and design.

There is a significant involvement of academic research groups and industry in measurement. CAIDA, AT&T, Sprint, Abilene, RIPE, Geant, France Telecom, and Intel have leveraged their ability to collect and analyze data about the operation of their own networks, compute traffic matrices, track traffic evolution and anomalies, and improve traffic measurement tools. In addition, large-scale cooperative testbed have been built for Internet measurement, *e.g.*, PlanetLab and EMUlab/Netbed. These testbeds lower the barrier to distribute experiment in network measurement. Furthermore, Sprint, AT&T and RIPE, many ISPs are building a community by funding researchers, making their traces available, and publishing their results.

1.1.1 Active *vs* Passive Measurement

There are two different measurement methods: passive measurement and active measurement. Passive measurements are commonly conducted by observing normal network traffic travelling through links or routers within a network. An example is counting the number of packets through a router in a period. Passive measurements do not inject extra probe packets into the network but rely on traffic flowing across the links or routers. Thus, the quality of passively gathered data depends on monitor placement, which requires the cooperation of network operators [56].

Active measurements are carried out by injecting probe packets into the network and observing their behavior. An example is the measurement of a traceroute path from a source to a destination. Some active measurement tools (such as those used for one way end-to-end delay measurements) require the cooperation of both end sites. Since active measurement imposes extra traffic into a network and can distort its behavior in the process, it may affect measurement results. Furthermore, some active probe

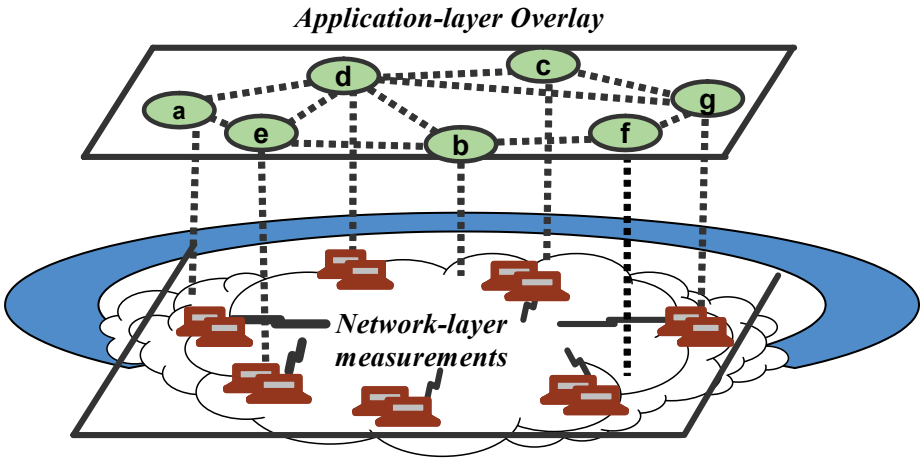


Figure 1.1: Overlay-layer measurement Vs. Network-layer measurement

measurements (such as ping) resemble denial of service attacks, so may be blocked by some ISPs.

Both active and passive measurement methodologies are useful in the collection of a large amount of data for the research purpose. However, several problems concerning collected data need be addressed:

(1) Data collection should not add a heavy network load, and when analyzing the measurement results, the effect of the network traffic itself must be taken into consideration.

(2) It is unclear how unbiased Internet measurement can be achieved.

(3) Identifying sufficient sampling rates is still unresolved.

(4) Some duplicate measurement can be avoided by cooperation and data sharing.

1.1.2 Network Layer *vs* Application Layer Measurement

Figure 1.1 shows measurement at the Network and Application layer. The Network layer refers to the network layer (layer 3) in the Open Systems Interconnection (OSI) model. This layer is responsible for end to end (*e.g.*, source to destination) packet delivery, and traffic problems management (*e.g.* switching, routing, and controlling the congestion of data). The main advantage of the Network layer measurements is that they can examine the various Networks performance at various Internet infrastructure components (*e.g.* routers, links, switches), and provide a "real check" of the delay, delay variation, loss, reordering, and throughput performance at the Network layer. Network layer measurements are often performed by ISPs and researchers to better monitor the traffic and Internet performance, as well as to improve design and add new services.

The Application layer refers to the application layer (layer 7) in the OSI model. Measurement at the Application layer is increasing. One main reason for this is its wide deployment. For example, a P2P network is a type of open, decentralized overlay network built on top of the Internet [74], on which distributed users communicate directly to find and share resources such as music and movie files. It has been shown that P2P accounts for more than 60% of the total Internet traffic [83]. In this kind of application-layer overlay network, the link virtually connects two nodes. These two nodes in the overlay may be far apart in the underlying IP topology and the virtual connection is realized as a path found by IP routers.

The main advantages of an application layer solution are the following: First, application layer application is easy to deploy and does not require changes at the network layer. An example is application-layer overlay multicast, which provides an attractive alternative to network layer multicast. The principle of building application-layer multicast is organizing nodes into data delivery trees on the application-layer multicast network, and reducing group communications to secure unicast communications. Second, the logical structure can easily be constructed to enable multicast trees to adjust to the dynamic underlying network conditions and nodes' behaviors in a scalable manner. Third, application-layer implementations can exploit the capabilities of lower layer protocols (such as TCP or UDP) in providing reliability, congestion control, flow control or security according to their needs.

Application layer measurements are needed and crucial to guarantee the performance quality of the applications. The prices to pay for the above advantages are reduced routing efficiency and overlay topology efficiency.

1.2 Internet Measurement Challenges

1.2.1 Network-layer Measurement Challenges

Network-layer Measurement challenges are mainly caused by several key properties of the Internet:

The first is that the Internet has vastly different policies and technologies. Its great success is largely a result of the main function of the TCP/IP protocol suite, which is the combination of different protocols at various layers [94]. TCP/IP allows vastly different networks administered by vastly different policies to inter operate seamless making it harder to understand precisely how a large IP network behaves.

The second key property is that the Internet is vast. It consisted of an estimated 350 million computers in July 2005 [47]. Its size causes two difficulties (according to V. Paxson and S. Floyd [80]). The first is that the range of heterogeneity mentioned above is very large, and the second is that large size also causes scaling problem: many networking protocols and mechanisms work fine for small and medium networks of less

than tens of thousands of computers, yet become impractical when the network is three orders of magnitude larger (today's Internet), not to mention possibly five orders of magnitude larger in the coming decade. Large scale results in extra complexity when maintaining critical network properties such as stability [86][101].

The third key property is that the Internet changes drastically over time. For example, started as a 4-link network in 1969, the Internet had an estimated 100 million in July 2000, and has now about 350 million computers [47], reflecting a growth of about 30% per year. Moreover, the growth rate of the Internet traffic is estimated to be close to 100 percent per year [70].

The fourth key property is that the different transition techniques and the lack of wide deployment of IPv6 make IPv6 measurements harder to understand. Over the last decade, IETF [46] has been working on the deployment of IPv6 to replace the current IPv4 protocol. The reason is that this will allow enlarging IP addresses, enhancing autoconfiguration, etc. One of the biggest challenges in the deployment of IPv6, however, is how to migrate IPv4-based infrastructures to those supporting IPv6. It is impractical and costly to replace existing IPv4-based networking infrastructures with IPv6. To ensure smooth and successful integration of IPv6 into existing networks, the IETF IPng Transition Working Group has been working on several different transition strategies, tools, and mechanisms to encapsulate IPv6 packets into IPv4 packets and transport them over an IPv4 network infrastructure.

1.2.2 Application-layer Measurement Challenges

Application layer measurement has several key properties that make it hard to analyze. The first is the highly dynamic nature of the application layer applications. For example, the nodes in the overlay network join and leave frequently. A recent study [83] found that more than 10% of the connections stay open for less than 10 seconds, while more than 60% of the connection stay open for less than 17 minutes.

The second key property is that the P2P layer is hidden [27][55]. Tracking only the application layer performance is not difficult. However, the drawback is that this does not allow observation of inside lower layers, so it is nearly impossible to explain behavior.

The third key property is that P2P data is hidden [55]. Identifying and measuring P2P traffic volumes across ISP networks is becoming an increasingly difficult activity [71][67]. Until comparatively recently, ISPs had to rely on simplistic port-based network reporting tools to measure the breakdown of traffic flows traversing the network. This approach may have proved reasonably accurate in the past but the growth in dynamic port usage and other stealth techniques employed by modern applications has rendered it extremely ineffective. These tools typically end up identifying a significant proportion of traffic as "unknown" or reporting false traffic levels for existing protocols.

The fourth key property is the number of P2P users is huge. The number of hosts

running on Gnutella was reported to be 2,219,539 hosts online at 11:56 AM, 21 April 2006 [107].

Internet measurement is relatively a new field. The results obtained in this field are very encouraging, but there are still many aspects of the Internet's structure, workload, and applications that are only poorly understood (*e.g.* a universal system for location; a new design for secure, robust network, operation in times of crisis, the emergence of different networks). At the same time, there are new challenges of Internet measurement concerning the computing and communications world, which might be materially different in 10 to 15 years time [21].

1.3 Thesis Objectives

Considering the diversity of research, it is important to precisely outline the contours of our work and to clearly formulate the scientific contribution of this thesis. Given different measurement methods and different researchers, ISPs, and large-scale cooperative testbeds involving in the Internet measure, many different "reality checks" are being performed to provide a means of exploring "real world" measurement, experimentation, simulation, and analysis. A standard measurement framework and metrics should therefore be well defined, and capable of being measured repeatedly and reliably. Hence,

Our first objectives in this thesis are

1. To present what is known about measurement's framework and metrics
2. To investigate the measurement's uncertainties and errors

The best-effort paradigm in the current Internet does not offer quality of service regarding the packets it transports, *i.e.* there are no guarantees with regard to the delay, jitter, loss, or reordering that packets experience, nor can it guarantee the bandwidth available along the traveled path. A novel aspect of our work is the use of real Internet measurement to evaluate the Internet performance.

Our research at the network layer measurement focuses on (1) hopcount and degree distributions; (2) reordering (*i.e.*, the out-of-order arrival of packets at the destination); and (3) the IPv6 delay and loss evolution.

(1) The interest in analyzing the hopcount measurements and degree distributions is that by combining their results, we will be able to better understand the infrastructure in current Internet.

The hopcount of a path is the number of nodes of that path. Measurements of the hopcount distribution are important to better understand the current topology and

to propose a more efficient network infrastructure than the current Internet. These results will also help simulate more realistic network topologies. Van Mieghem *et al.* [102][103] have shown that the hopcount distribution of the Internet is well modeled by that of a random graph with uniformly or exponentially link weights. One focus of this thesis is to compare our measurement results with the work of Van Mieghem *et al.*. We also studied the change of the hopcount distributions over time as they may indicate dynamic changes in the Internet.

The degree of a node is the total number of its neighbors connected to it. Measurement of the degree distribution is useful to determine important global characteristics of the Internet structure, and is frequently used to simulate realistic network topologies. Faloutsos *et al.* [34] have shown that the degree distribution of the Internet follows a power law, and found that Internet models before [34] failed to exhibit power laws. This result had significant impact on network topology research [19][97][61][14][52]. It is thus interesting to compare our measurement results with the works of Faloutsos *et al.*.

(2) Reordering is a phenomenon in the Internet [75][8], and frequently occurs on the high-speed links. The major cause of reordering has been found to be parallelism in Internet components (switches) and links [8]. For example, due to load balancing in a router, the packets of a same stream may traverse different routers, with each packet experiencing a different propagation delay, and thus may arrive at the destination out-of-order. Reordering may also be caused by the configuration of the hardware (*i.e.*, multiple switches in a router) and software (*i.e.*, class-based scheduling or priority queueing) in the routers.

The interest in analyzing end-to-end reordering is that reordering impacts greatly on the performance of applications in the Internet. In a TCP connection, the reordering of three or more packet positions within a flow may cause fast retransmission and fast recovery multiple times resulting in a reduced TCP window and consequently in a drop in link utilization and, hence, less throughput for the application [59]. For delay-based real-time service in UDP (such as VoIP or video conference), the ability to restore order at the destination is likely to have finite limits. The deployment of a real-time service necessitates certain reordering constraints to be met. For example, in the case of VoIP, maintaining the high quality of voice requests that packets be received in order, and also within 150 milliseconds. To verify whether these QoS requirements can be satisfied, knowledge about the reordering behavior in the Internet appears desirable.

(3) IPv6 is currently moving continuously towards commercial deployment, and its performance knowledge could be crucial for ISPs to understand how to provide a high quality IPv6 network for future Internet applications. Although packet delay and loss are two important parameters of the Internet performance, to the best of our knowledge, the evolution of large-scale IPv6 [28] delay and loss performance has not been previously studied. Qualitative evaluation of IPv6 performance requires measurements collected over time, combined with information about delay, path, and tunnel discovery.

Earlier studies focused mainly on IPv6 transition technologies [2] or identifying IPv6 network problems in a dual-stack world by using measurements from only a few days [20][105]. Compared to IPv4, IPv6 is still in its infancy and is rarely used by real-life applications. There is a lack of knowledge about the network performance of end-to-end IPv6 communication. Therefore, studying the large-scale IPv6 delay and loss performance evolution is important to understand the performance of the current IPv6 networks, and to provide high quality services for future Internet applications. To summarize,

Our objectives with respect to Network-layer measurement are

1. To investigate the hopcount and degree distributions of the Internet
2. To evaluate to what extent reordering can impact the application
3. To qualitatively evaluate the delay and loss evolution of IPv6 Network

To end-users, it is not the network performance that matters most but the perception of the quality of applications running over the network. In general it can be stated that the large scale deployment of applications will only be successful if the perceived quality of these applications is sufficiently high.

VoIP and P2P are two applications that have recently attracted a lot of attention due to their wide deployment. Our researches on the application layer measurement are therefore focused on the (1) assessment of VoIP quality; and (2) P2P distance estimation methods.

(1) Assessment of VoIP quality. Some methods have also been proposed for estimation of VoIP perceived quality. The idea is to quantify the effect of individual impairments on conversation quality, then quantify Network-layer performance in terms of delay and loss statistics [65][108][64]. The purpose is to map those measurable metrics to the user's opinion score. Because interactive services such as VoIP are not only increasingly important but also impose stringent requirements on the network, assessing the performance of VoIP is an important issue. Many Internet operators offer services to small and medium enterprises. They provide email, Internet access with a firewall, Windows networking and backup services, as well as national/international VoIP. It is essential for these ISPs to understand how different performance factors (*i.e.* different delay, packet reordering, and packet loss) affect the perceived quality of voice calls.

(2) P2P distance estimation methods. A number of methods have been proposed for estimation of network delays based on reduced or incomplete measurements [72][42][81][26]. A promising method that has received considerable attention is assigning coordinates to nodes. The idea is to assign coordinates in such a manner that the associated distance approximates network delay (round-trip propagation and transmission time). An

efficient mechanism to estimate distance in the Internet may be useful for many large-scale distributed network applications such as nearby server selection and peer-to-peer computing. An example is a client selecting the nearest from a set of equivalent servers. Similarly, optimization of overlay networks like peer-to-peer networks often requires that nodes connect to peers in their neighborhood. To summarize:

Our objectives with respect to Application-layer measurement are

1. To assess the current VoIP perceived quality of the Internet
2. To evaluate a fixed landmark-based distance estimation scheme

1.4 Thesis Outline

The organization of this thesis is schematically depicted in Figure 1.2. It consists of three parts. The first part presents and discusses what is known about the measurement framework and metrics, as well as measurement uncertainties and errors; the second part is dedicated to the measurement at the Network (IP) layer, and the third part to measurement at the Application layer.

Chapter 1 describes the problems under consideration, defines the notation used, and discusses our motivation and research objectives.

The first aim of the thesis is to present and discuss what is known about the measurement framework and metrics, as well as measurement uncertainties and errors. Chapter 2, therefore, provides background material to provide a basis for evaluating the performance of different Internet components, such as one-way end-to-end active measurement framework, and the measurement metrics (*e.g.* connectivity, delay, delay variation, loss and reordering). It also discusses the measurement uncertainties and errors, and the measurement projects used in this thesis.

After clarifying the background material we deal with the second aim of our thesis in Chapter 3, 4 and 5; the use of Network-layer measurement to evaluate Internet performance.

Chapter 3 analyzes the connectivity properties of Internet. The networks can be modeled as graphs consisting of nodes connected by links. To better understand the structural properties of the Internet, two important parameters, hopcount distribution and degree distributions, are studied. The key research questions are how to model hopcount distribution and degree distribution through the real measurement. The conclusions indicate that (1) the hopcount distribution in the Internet can be well modeled by that of a random graph with uniformly or exponentially distributed link weights, and (2) for large group sizes, the node degree distribution appears to obey a

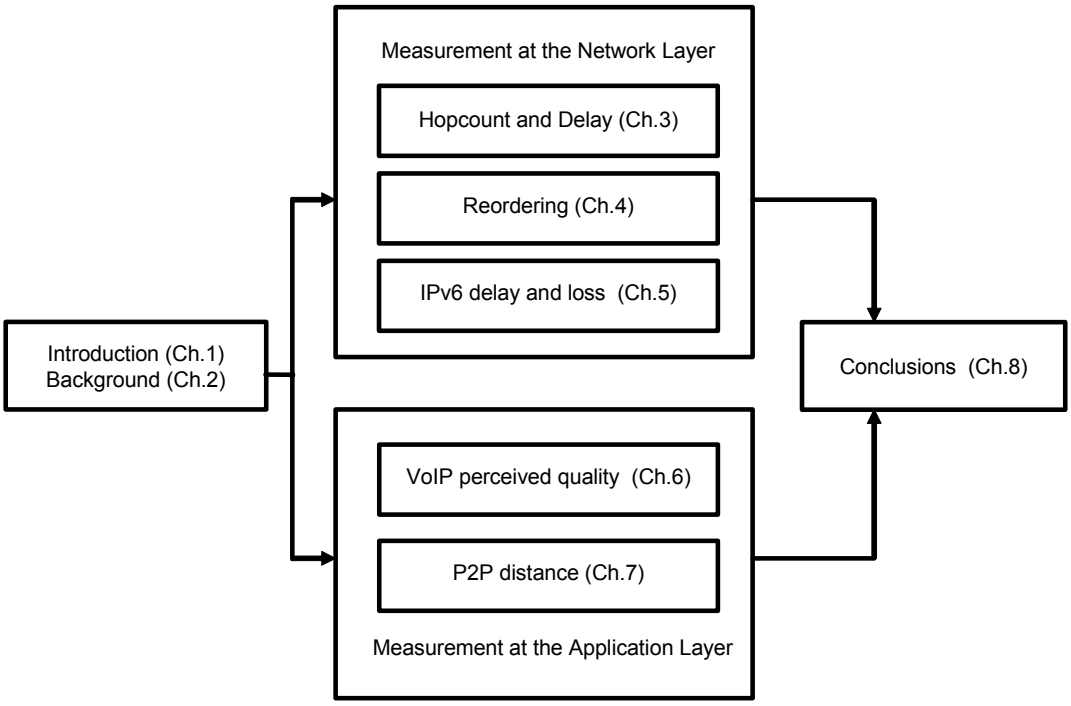


Figure 1.2: Overview of the thesis outline

power-law, while for small group size, the node degree distribution appears better fitted with exponential distribution.

Chapter 4 analyzes reordering in the Internet. The key research questions are how to define reordering in measurement, and to what extent reordering can impact on the application. The conclusions indicate that reordering impacts greatly on the performance of applications in the Internet, and the extent of reordering depends on the network load.

Chapter 5 analyzes IPv6 delay and loss performance evolution. The key research question is how to qualitatively evaluate the IPv6 infrastructure. We answer the above question by comparing the IPv6 and IPv4 delay and loss evolution measurements under the current network situations. We further focus on problems present in the IPv6 paths, and run traceroutes with path MTU (Maximum Transmission Unit) discovery to identify the causes.

The third and final aim of the thesis is to evaluate user application (*e.g.* VoIP and P2P) performance through Internet measurement. Chapter 6 therefore evaluates VoIP perceived quality. The key research question is whether the current IPv4 Internet can achieve a high VoIP perceived quality. Chapter 7 evaluates a fixed landmark-based estimation scheme using real measurement data for the delay and hopcount between Internet hosts.

Finally, Chapter 8 concludes our work, and shows that accurate measurements not only enhance our understanding of the current Internet, but can also lead to recommendations for improvements on both the network infrastructure and network protocols.

Chapter 2

Internet Measurements

2.1 One-way End-to-End Active Measurement Metrics

2.1.1 The Reason for Active Measurement

Passive measurement and active measurement are two different measurement methods. Passive measurements are carried out by observing normal network traffic. They are commonly used to measure traffic flows, *e.g.*, counting the number of packets and volumes traveling through links or routers within a network. Passive measurements do not add extra probe packets into the network but rely on traffic flowing across the links or routers.

Active measurements are carried out by sending probe packets into the network. This thesis investigates active measurement because of its many advantages. These include: (1) Ability to generate traffic between selected nodes. (2) Flexibility to design probe streams with particular properties to match measurement requirements. For example, different probe packets with different protocols can be sent into the network to measure different metrics, varying from average delay and loss on a route, to reordering, bottleneck and available bandwidth in the Internet. (3) Other advantages include an enormously reduced volume of measurement data compared to the passive monitoring of high bandwidth links, and the avoidance of data privacy issues.

2.1.2 The Reason for One-way Measurement

The Internet path from a source to a destination may be different from the path from the destination back to the source. In a quality-of-service (QoS) enabled network, therefore, the QoS guarantee provided in one direction may radically differ from that in the reverse direction. Measuring the paths independently allows verification of both

QoS guarantees.

2.1.3 The IPPM Framework for Metric Definitions

To achieve an accurate common understanding of the Internet performance and reliability between users and ISPs, before the end of May 1999, the IETF's IP Performance Metrics (IPPM) Working Group has developed a measurement framework (*RFC 2330* [78]). The Framework presents terms for describing networks, explains the need for metrics to be useful, understood, concrete, well defined, and capable of being measured repeatedly and reliably. The purpose of *RFC 2330* is to define a general framework for particular metrics to be developed by the group. For example, when talking about the measurement, we need to define the exact type of traffic, the payload being measured (*e.g.* protocol number, UDP or TCP port number, size, and precedence). Examples include:

- Internet vocabulary about Internet components such as routers, paths, and clouds are defined. For example, it defines a term "Path"as a sequence of the form $< h_0, l_1, h_1, ..., l_n, h_n >$, where $n \geq 0$, each h_i $(0 \leq i \leq n)$ is a host which is a computer capable of communicating using the Internet protocols, each l_i is a link between h_{i-1} and h_i. A pair $< l_i, h_i >$ is termed a "hop". In an appropriate operational configuration, the links and routers in the path facilitate network-layer communication of packets from h_0 to h_n. Note that path is a unidirectional concept.

- Each metric will be defined in terms of international standard units of measurement. For example, a time will be expressed in UTC, and the unit of information is the bit.

- Those who develop the measurement methodologies should try to understand the sources of uncertainty and errors, and quantify the amounts of uncertainty and errors, and minimize their uncertainty and errors.

- How to achieve an unbiased sampling method remains unsolved (see Section 2.2.1).

In general, notions defined in *RFC 2330* can be the base for defining measurement metrics.

2.1.4 One-way Packet Connectivity

Connectivity is the basis of the Internet. Understanding it is helpful in understanding Internet infrastructure.

IPPM metrics for measuring connectivity (*RFC 2498* [63]) defines a series of metrics for connectivity between a pair of Internet hosts (IP addresses). It builds on the notions introduced and discussed in *RFC 2330*. If a packet transmitted from Src (source) to Dst (destination) at time T arrives at Dst, then Src (the IP address of a host) has connectivity to Dst (the IP address of a host) at time T. Note that T is not explicitly defined since there is propagation or processing delay between any path. In theory, the TTL field in IP packet header limits packet lifetimes to 255 seconds (*RFC 791*), while in practice, the TTL field can be a strict hop count, with most Internet hops being much shorter than 1 second. Src has connectivity to Dst during the interval $[T, T + \Delta T]$ if for some $T^{'}$ within $[T, T + \Delta T]$ it has connectivity to Dst.

2.1.5 One-way Packet Delay

Network delay directly influences the user experience in many applications. Some applications do not perform well if end-to-end delay between hosts is large relative to some threshold value.

One-way packet delay consists mainly of propagation delay, processing delay, transmission delay, and queueing delay. Propagation delay is the time taken by a transmitted bit to travel from one end of a link to the other end, and is only dependent on the speed at which signals travel on the transmission medium (roughly 5μs/km) and the length of the link. Processing delay includes time to lookup the routing table and to move the packet over the switch fabric. Transmission delay is the amount of time required by the router to push out the entire packet onto the link. Queueing delay is the waiting time in the output buffers (input queue in some cases).

The packet has a One-way delay (*RFC 2679* [5]) ΔT ($\Delta T > 0$) from Src to Dst if that Src sent the first bit of a packet to Dst at time T and that Dst received the last bit of the packet at time $T + \Delta T$. The minimum delay indicates the propagation and transmission delay, and also indicates the delay likely to be experienced in the slightly loaded path. Delay above the minimum indicates congestion present in the path. If the packet fails to arrive within a reasonable period of time (such as 255 seconds), one-way delay is taken to be undefined. In the measurement application, the packet gets its transmission timestamp just before it is sent on the socket by the application and transmitted to the network interface card (NIC). If the packet arrives within a reasonable period of time, the application takes the arriving timestamp from the kernel. By subtracting the two timestamps, an estimate of the one-way delay can be computed.

GPS system has a measurement accuracy of about 10 μsec, while the Network Time Protocol (NTP) only has a measurement accuracy of several $m$sec. Since delay values often can be as low as on the order of 100 μs (to 10 ms), it is important for Src and Dst to synchronize their clocks precisely with the GPS system. Uncertainty in these values (*e.g.* systematic error, see Section 2.2.2) must be taken into account in error analysis

which is an important part of the analysis.

2.1.6 One-way Packet Delay Variation

The uses of delay variation include determining the size of play-out buffers for real-time applications (such voice or video over IP), and determining the dynamics of queues within a network.

The delay variation may be due to load balancing in a router, the packets from a source to a destination may traverse different routers, where each packet experiences different propagation delay, and thus may arrive at the destination with different packet delay. It may also be due to packets suffering different queue delays within a router. In addition, this metric is sensitive to differences and variations of the clocks of the two hosts.

One-way packet delay variation $d\Delta T$ (*RFC 3393* [29]) means that Src sent two packets, the first bit of the first packet at time $T1$, and the first bit of the second packet at time $T2$; and the last bit of the first packet was received by Dst at time $\Delta T1 + T1$, and at time $\Delta T2 + T2$ for the second packet, and that $d\Delta T = \Delta T2 - \Delta T1$.

2.1.7 One-way Packet Loss

The one-way packet loss (*RFC 2679* [5]) from Src to Dst at T is defined as 0 if Src sent the first bit of a packet to Dst at time T and Dst received that packet. One-way packet-loss is exactly zero when the one-way delay is a finite value, and 1 when the one-way delay is undefined. Packet loss occurs where network traffic fails to reach its destination within a reasonable period of time. This may be due to network congestion, or the change in a source-destination path. For example, in the case of traffic congestion, network devices like switches and routers have to buffer packets in their queues when a link is congested. If the link remains congested for too long, the queues will overflow and packets will be dropped.

Understanding one-way packet loss from a source to a destination is useful to understanding dynamic Internet performance. Real-time applications and transport-layer protocols to sustain high bandwidths are sensitive to loss. For example, some real-time applications do not perform well if loss between hosts is large relative to a threshold value; and it is difficult for transport-layer protocols to sustain high bandwidths when the packet loss is large.

2.1.8 One-way Packet Reordering

Reordering is the out-of-order arrival of packets at the destination [75][8]. In practice, a packet is classified as a reordered or out-of-order packet if it has a sequence number smaller than its predecessors at the destination.

Reordering may be caused by parallelism in Internet components (switches) and links [8]. For example, due to load balancing in a router, the packets of a same stream may traverse different routers, where each packet experiences a different propagation delay, and thus may arrive at the destination out-of-order. Reordering may also be caused by the configuration of the hardware (*i.e.*, multiple switches in a router) and software (i.e., class-based scheduling or priority queueing) in the routers.

A reordering metric has an impact for most real-time applications, such as VoIP and video conferencing. Currently there is no good metric to qualify the extent of the reordering. The extent of reordering may be sufficient to cause a received packet to be discarded by functions above the IP layer. More details will be shown in Chapter 4.1.

2.2 Uncertainties and Errors in the Measurement

The following sub-sections describe three crucial issues with regard to measurement errors and uncertainties that, in general, cannot be avoided. To make more accurate measurement, these uncertainties and errors in the measurement must be quantified.

2.2.1 Uncertainties and Errors in Sampling

The traffic measurement database should contain at least the following attributes of the packets: Timestamp (with sufficient accuracy), total packet size, source and destination address (*e.g.*, network level addresses), packet sequence (*e.g.*, IP protocol sequence number), and protocols in the packet (plus some important protocol parameters, *e.g.*, TCP flags to distinguish between data and acknowledgement packets).

For accurate traffic trace (full measurements), every packet must be registered at every measurement point. This method is not feasible in reality due to the size of measuring probes. It is, however, possible to sample some probe packets in the network and estimate accurate delays, jitters, throughputs and losses between measurement points. When assessing variations based on a sample, it is generally assumed that the sample is unbiased. Unfortunately, Lakhina *et al.* [57] have recently pointed out that as a tool for measuring degree distribution, traceroute sampling has a more fundamental bias. The papers demonstrate a systematic bias in the measurement technique used to gather the data for the 1999 Internet study–pronounced enough to make even classic random graphs look heavy-tailed, therefore the conclusions made by such studies (*e.g.* power law nature of degree distribution) may not reflect reality. How to achieve an unbiased sampling method remains unsolved.

Two common ways are periodic sampling and random additive sampling. Periodic sampling is easily anticipated but is biased since it can drive a network into a synchronization state [36] and greatly magnifies minor effects. Random additive sampling is more unbiased, and randomly injects data with a common statistical distribution $G(t)$

[9]. One popular method is Poisson sampling if $G(t) = 1 - e^{-\lambda t}$, where rate $\lambda > 0$ is an integer-valued. Note that if λ is too small, there will be not enough interesting network behavior, however, if λ is too large, measurement traffic can cause congestion.

2.2.2 Uncertainties and Errors in Delay Measurement

Since delay values will often be as low as the 100 μs to 10 ms range, it is important for Src and Dst to synchronize the clock very closely with GPS systems (with a measurement accuracy of 10s of μsec) rather than NTP (with a measurement accuracy of several $m$sec).

There are four main reasons for clock uncertainty (*RFC 1305* [66]):

- Synchronization (*i.e.* the extent to which two clocks agree on what time it is)
- Accuracy (*i.e.* the extent to which a given clock agrees with UTC)
- Resolution (*i.e.* the precision of a given clock)
- Skew (*i.e.* the change of accuracy or synchronization with time).

Keeping this in mind, the measured value is calculated as:

$$\text{measured value=true value + systematic error + random error} \tag{2.1}$$

Both the systematic error and random error are generated by the instruments themselves. Uncertainty in these values must be taken into account in error analysis of a given implementation of the method.

Observing TTM [87] delay distribution between a random Src-Dst path over a day, we have discovered that the heavy tail did not monotonically decrease to 0, and some outliers could represent the system errors. To better understand the sources of uncertainty or error infects on the sending and receiving sides, as well as to further quantify the amounts of uncertainty or error to collect sufficient statistics, we ran a test measurement to verify delay accuracy in the lab using an even simpler setup (Figure 2.1): 2 test-boxes connected back-to-back with a one-meter-long cross cable over Ethernet. Packets were sent from a measurement PC (Pentium III) running FreeBSD to a similar PC using 100 Mbps Ethernet. The packets were time-stamped twice: first on the sending side, as the last action (in application layer) before the packet was written on the socket, and then on the arriving PC as soon as the packet was released to the operating system (OS) by the Ethernet card (data link layer). This approach attempts to ensure that only a minimal amount of time elapses between time-stamping and transmission of the packet on the wire.

With the aid of the GPS (global position system), this measurement system promises an accuracy of around 10 μs. In order to be able to distinguish between the effects on the

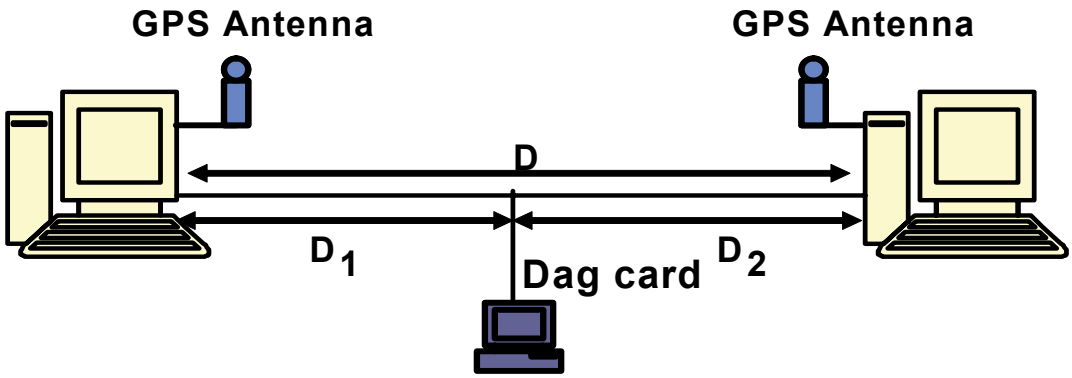


Figure 2.1: Overview of the setup of the measurement

sending and receiving side of the test-setup, a DAG-card [41] with measuring accuracy of about 500 ns was installed between the two machines. This card allows monitoring of the packets travelling on the Ethernet cable and thus separates effects on the sending and receiving side. The difference between the time-stamps of departure at the source and arrival at the destination box is termed the total delay D, while the difference between the time-stamps of departure at the source and DAG card at the middle is termed D_1. Finally, the delay from the middle of the link to the receiving box is termed D_2.

Probe-packets were sent over the direct link between two test-boxes at rates of 1 packet per second (ps), 10 ps, and 100 ps respectively, above the original design value of around 1 packet per 30 seconds in RIPE TTM. The probe packets were 100 bytes long, and contain a UDP frame with destination port 8000. The interval between two consequent packets is selected at random, with intervals according to a Poisson process, as discussed in [36] and (*RFC 2330* [78]). A total of about 30,000, 300,000 and 500,000 probe-packets were transmitted at the successive rates, respectively. In theory, D_1, representing the propagation delay of a packet through a link of 1 meter length, is about $5 \cdot 10^{-3}$ μs, and D_2, representing the transmission delay, is about 8 μs, both are negligible. The resulting two delays D_1 and D_2 therefore show the processing delays of the test boxes.

Those results of 1 ps are shown in Figure 2.2 (The results of 10 ps and 100 ps show the similar behaviors). At rate 1 ps the 1-CDF plot (Figure 2.2) lines up to the D_1 value of 130 μs, which is the 97.98% empirical quantile [7]. For the rates 10 ps and 100 ps, these value are 133 μs and 99.65%, 120 μs and 99.18% respectively.

D_2 (at all three rates) has small interquartile range and a large standard deviation. It has a peaked histogram with a long tail and hence cannot be described by normal distribution. The experimental results in Figure 2.2 confirm that the values of D_2 larger

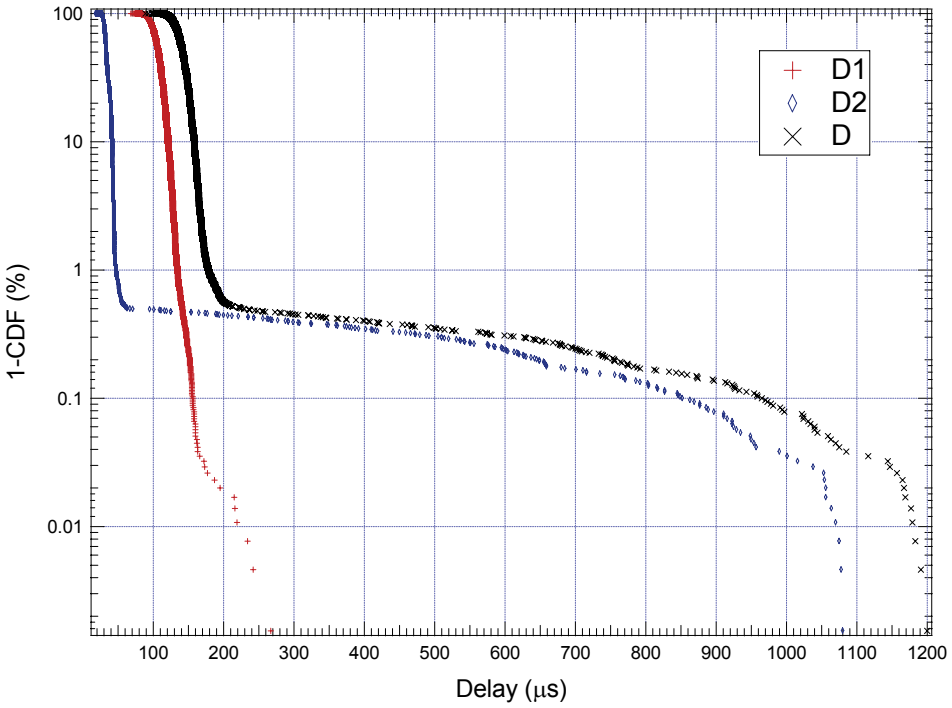


Figure 2.2: Delay measurement accuracy

than 63 μs, 80 μs and 85 μs for different rates show uniform behavior. The fraction of those values is 0.5%, 0.2% and 0.55% respectively.

Of course, it is the total delay D caused by the measurement process that is of interest. Since D_2 has a heavier tail than D_1, and D is the sum of D_1 and D_2, one expects D to have the same tail behavior as D_2. D clearly has a uniform tail. More precisely, the values of D larger than 230 μs, representing 0.5% of the data, are uniformly distributed. D also has a uniform tail for the rates 10 ps and 100 ps (from 300 and 240 μs).

One possible reason for the heavy tail of D_2 is that when NIC sends an interrupt to OS for the arrival packets, the OS may be busy with other computing processes with higher priorities, and respond later to the interrupt request, thus adding extra delay. Since minimal, average, and maximal delay are sensitive to the clock error, to minimize this error for each Src-Dst path, 2.5 percentile, median and 97.5 percentile delay are a better choice. For completeness, histograms' mean and RMS (root-mean-square or standard deviation) can also be provided.

2.2.3 Uncertainties and Errors in Traceroute Measurement

Traceroute measurement [94] is widely used to detect and diagnose routing problems, investigate end-to-end paths through the Internet, and discover the underlying network topology. This tool identifies the interfaces on a forwarding path and reports round-trip time statistics for each hop along the way. Despite its many well-documented limitations, it is an effective way of determining how packets flow through the Internet and is useful for network operators in identifying forwarding loops, black holes, routing changes, unexpected paths through the Internet and, in some cases, the main components of end-to-end latency. Researchers rely heavily on traceroute to study routing protocol behavior [92], network performance [76], and the Internet topology [54].

Some fundamental limitations of a traceroute can lead to erroneous data. These include:

1. The router encountered in the path of a traceroute may not have a corresponding host name registered in the Domain Name Server. Beside, as the traceroute tool sends 3 probes for each traceroute paths, each probe may take a different path due to changes in route caused by fluttering. This is primarily the effect of load balancing. Traceroutes also generate "*" messages that were filtered out as erroneous data. The "*" messages only indicate that the host is unreachable or the network is unreachable and have no significance in towards measuring network parameters.

2. A significant problem using traceroutes is the alias problem. The traceroute returns all the source addresses of the "Time exceeded" ICMP messages. These addresses represent the interfaces on the routers that received traceroute probe packets. Thus a traceroute can not determine which interface IP addresses belong to the same router.

A study of which traceroute errors users observe in the current Internet routing will be presented in Section 3.2.

2.2.4 Other Uncertainties and Errors

For accurate measurements analysis, the effect of the network traffic itself must be included. High traffic load on the measured network creates extra queuing delays, which may result in false information about the real network performance. For example, reordering has been shown to depend on the network load [109]. It is necessary, therefore, to be aware of current load of the network when performing measurements. Otherwise no accurate decisions can be made on the basis of the measurements.

2.3 Measurement Projects

Table 2.1 compares characteristics of several public Internet measurement infrastructure projects open to Internet researchers.

Projects	Active/Passive	Analysis	Monitors
RIPE NCC	Both	Topology/routing	80+
CAIDA skitter	Active	Topology/performance	20+
PlanetLab	Both	Topology/routing/performance	600+
Surveyor	Active	Topology/performance	50+
Route Views	Passive	Topology/routing	40+

Table 2.1: Research Measurement Infrastructures

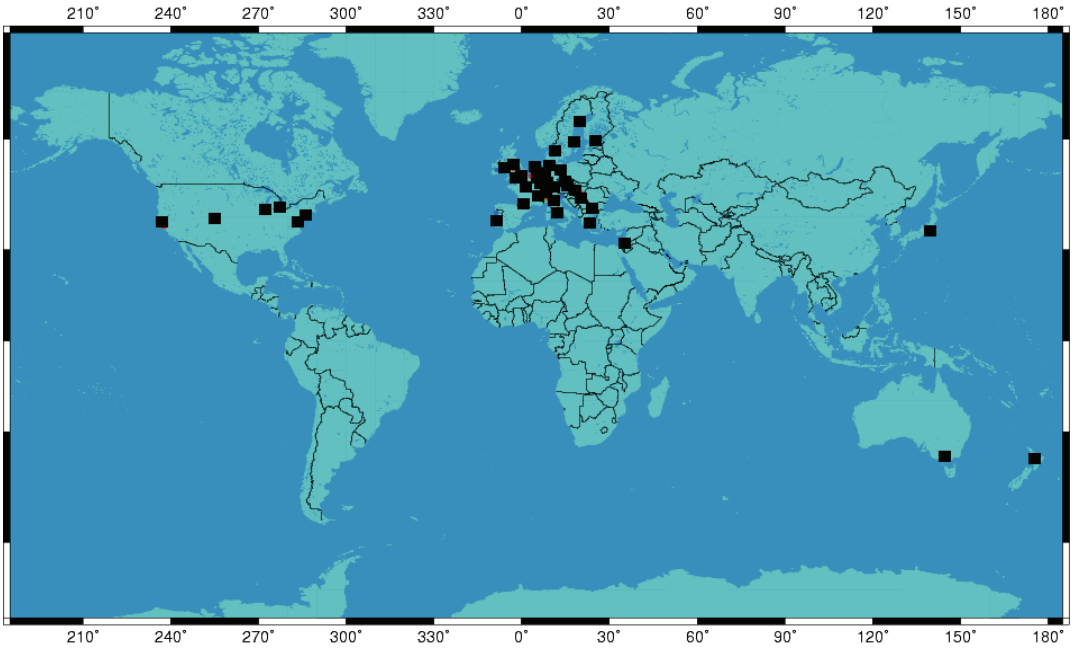


Figure 2.3: Location of the RIPE test-boxes

RIPE NCC [87] is a collaborative organization open to groups and individuals in Europe and beyond. The object of the RIPE TTM project is to collect Internet routing and topology information. TTM actively measures one-way delays and losses by sending time-stamped packets to each testbox, and it collects routing information by collecting multiple traceroute paths.

CAIDA's skitter [15] actively collects topology and performance data from approximately two dozens monitors to hundreds of thousands of destinations in IPv4 address space. The object of skitter is to analyze topology and performance of the Internet.

PlanetLab [82] is a group of computers available as a testbed for networking and distributed systems research. It was established in 2002 and consisted of 642 nodes in 304 sites worldwide in January 2006. Each research project has a "slice", or virtual

machine access to a subset of the nodes. PlanetLab members actively participate in developing tools for the greater community and, as a result, each user has a wide choice of tools to complete regular slice maintenance tasks.

Surveyor [95] was a project to perform Internet measurement world-wide. This project was also developing methodologies and tools to analyze collected performance data (such as delay).

Route Views [89] is a project to obtain real-time information about the global Internet routing information collected in several backbones and locations.

Since we used the data of RIPE TTM, CAIDA skitter and PlanetLab data to evaluate the Internet performance, we will explain these three projects in more detail.

2.3.1 RIPE NCC TTM

The TTM infrastructure [87] consists of approximately 80+ IPv4 and 30+ IPv6 measurement boxes scattered over Europe, Asia and USA shown as Figure 2.3. As RIPE NCC is connected to the Amsterdam Exchange Point, it maintains many IPv6 peers with other 6net participants, using BGP4+ as Exterior Gateway Protocol (EGP). As shown in Figure 2.4, between each path of measurement boxes, both IPv6 and IPv4 UDP packets of a fixed payload (100 bytes), called probe-packets, are continuously transmitted with interarrival times of about 30 seconds, resulting in a total of about 2886 probe-packets between each path per day. The sending measurement box generates an accurate time-stamp synchronized via GPS in each probe-packet, while the receiving measurement box reads the GPS-time of the arrival of the probe-packet. The end-to-end delay is defined as the difference between these two time-stamps and has an accuracy of about 10 μs. The hopcount of a path between two measurement boxes is measured about every 6 minutes using traceroute. In order to collect the information from both senders and receivers, and for easy access from the same place to the data files, there is a central point in RIPE NCC which collects all the traceroutes from each source-destination paths.

Then, as shown in Figure 2.5, the traceroutes are inserted into 2 tables (see table 1 and table 2) in a MySQL database[1] in a local computer at Delft University of Technology, the Netherlands. The MySQL database is chosen for easy performance. The first table, an example of which is shown in table 1, contains a routeid (each unique combination of IP-addresses seen between the source testbox and the destination testbox is mapped to a unique routeid) and the count (the number of the routeid recorded by the database). The second table (some examples are shown in table 2) contains all the IP-addresses of each router that appeared in the routing vector (routeid), and the length of the traceroute. The traceroutes between the boxes can be queried through the MySQL database. Here, table 1 shows all the traceroute records output between two

[1] www.mysql.com

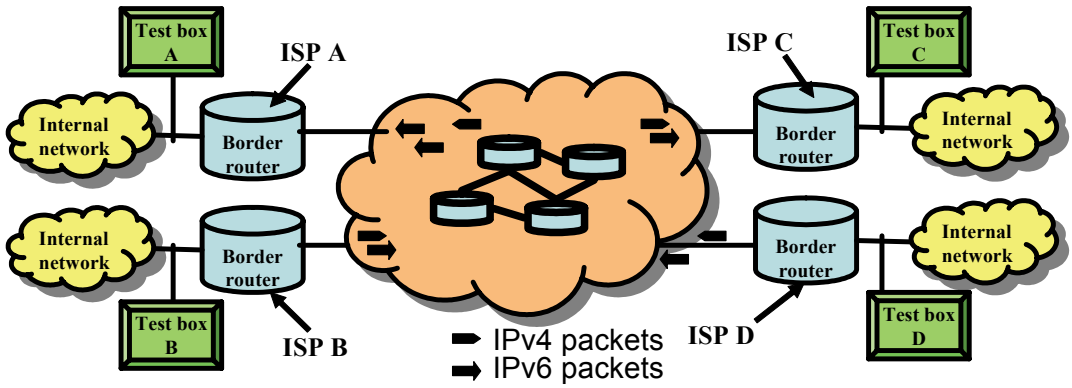


Figure 2.4: The RIPE TTM experimental setup

mysql > SELECT routeid ,COUNT(*) AS count FROM Records WHERE src =1 AND dst =18 GROUP BY routeid ORDER BY count DESC;

routeid	count
5	1255
25	1250
63712	671
63679	670
7980	541
19503	540
1290	40
16017	10
62260	8
62262	7
1281	1
140632	1
1298	1
1304	1
1348	1
16015	1
16550	1
32920	1
131678	1

19 rows in set (0.11 sec)

Table 1

mysql > SELECT len ,route FROM Routes WHERE ID= 5;

len	route
5	3238002702 3238002738 3238002932 3247705900 3285074646

1 row in set (0.04 sec)

mysql > SELECT len ,route FROM Routes WHERE ID= 25;

len	route
5	3238002702 3238002742 3238002932 3247705900 3285074646

1 row in set (0.00 sec)

mysql > SELECT len ,route FROM Routes WHERE ID= 63712;

len	route
10	3238002702 3238002738 3238002932 3247705890 2435385229 2435383377 3556884657 3556884994 3285074666 3285074646

1 row in set (0.00 sec)

Table 2

Figure 2.5: Tables in the MySQL database

Table 3

```
#ifndef __CINT__
#include "TROOT.h"
#include "TApplication.h"
#include "TCanvas.h"
#include "TChain.h"
#include "TTree.h"
#include "TH1.h" #include "TH2.h"

#include "Delay.h"


#endif
main () {
  ...
}
```

Table 4

1	**Src**Identifier for the sending test-box
2	**Dst**Identifier for the destination test-box
3	**Delay**Delay (ms)
4	**Nhops**Number of hops
5	**Route**Identifier for the route the packet took
6	**SrcErr**Estimated error on the sending clock
7	**DstErr**Estimated error on the receiving clock

Table 5

```
1 90 8.997 12 3776020 0.003 0.001 0
1 90 8.673 12 3776020 0.003 0.001 0
1 98 0.293 3 2953970 0.003 0.013 0
1 90 9.291 12 3776020 0.003 0.001 0
1 97 0.305 3 2735659 0.003 0.011 0
1 93 17.286 14 3947956 0.003 0.021 0
1 91 13.311 13 3776024 0.003 0.003 0
1 90 8.853 12 3776020 0.003 0.001 0
1 90 8.727 12 3776020 0.003 0.001 0
1 89 5.226 11 4409257 0.003 0.004 0
1 90 9.121 12 3776020 0.003 0.001 0
1 88 43.765 9 4406846 0.003 0.08 0
1 87 70.215 14 4178204 0.003 0.006 0
```

Figure 2.6: Tables in the ROOT database

text-boxes, and route ID 5 presents one special series of IP addresses shown as table 2.

The traceroute can enhance our view of the structure of Internet. The changes of traceroute (say, caused by load balance) can explain how IP packets were transferred between two points is suddenly changed. It also can accurately tell how the Internet communication changes.

RIPE TTM uses CERN's ROOT [18] package to process and store all test traffic delay data. ROOT provides an object oriented data analysis framework, featuring graphics, histograms, a C++ interpreter and object I/O. This provides a convenient environment for (interactive) analysis and data presentation.

To allow easy combination of multiple files into one bigger data set, the test traffic data are stored in TTree objects. Each TTree has a collection (usually one day) of delay measurements. Having verified ROOT installation to function properly, one is ready to create the support library for TTM analysis. The delay measurement are stored in objects of a class Delay, which has been defined by TTM[2]. This class definition and its member functions have to be made known to ROOT by dynamically loading a shared library.

Next, as shown in the table 3 of Figure 2.6, the last file (Delay.h) contains the definitions of the TTM Delay class, the others are from ROOT itself. Enclosing the statements in a #ifndef CINT clause allows the same code to be called either as a macro or compiled as a standalone program.

When extracting ROOT data, a table with data can be created and imported into

[2]The source code for the Delay class library can be loaded from our ftp site: ftp://ftp.ripe.net/test-traffic/ROOT/libDelay. After the delay library is created, the TTM analysis software is ready for use.

Hop	IPv6 address	AS	MTU
0	ttxx		
1	xxx:0:9::1	559	1500
2	xxx:0:c02e::2	*N/A*	1500
3	xxx:0:c03c::1	559	1500
4	xxx:0:c006::1	559	1500
5	xxx:12:10aa::1	20965	1500
6	xxx:1201:1e01::2	20965	1500
7	xxx:1001:1e01::1	20965	1500
8	xxx:10dd::2	20965	1476
9	xxx:ff:1::2	1853	1476

Figure 2.7: IPv6 traceroute and tunnel discovery results

other applications. The data, example of which are shown in table 4 of Figure 2.6, consists mainly of 7 columns. The fields are: (1) Src: Identifier for the sending testbox, (2) Dst: Identifier of the destination testbox, (3) Delay: end-to-end delay in ms, (4) Number of hopcounts, (5) Route: Identifier for the route the packet took, (6) SrcErr: Estimated error on the sending clock, and (7) DstErr: Estimated error on the receiving clock. Traceroute records of the route Identifier for the route the packet took can be queried through the MySQL database as described in Table 2 of Figure 2.5. An example of ROOT's output is shown in the table 5 of Figure 2.6. This ROOT database is also saved in a local computer at Delft University of Technology.

IPv6 paths are periodically collected with traceroute6. In addition to the traceroute measurements, TTM uses the tunnel discovery tool [24] to identify tunnels in those collected IPv6 paths once per hour by measuring the Maximum Transmission Unit size (normally 1500) over an entire path, since a drop in MTU at an intermediate router indicates a possible tunnel entry point. An output example of the MTU record from a source located in Zurich, CH, to a destination located Vienna, AT, is shown in Figure 2.7, where a returned value of "1500" and "1476" indicates a native path and Generic Routing Encapsulation tunnels [43] respectively.

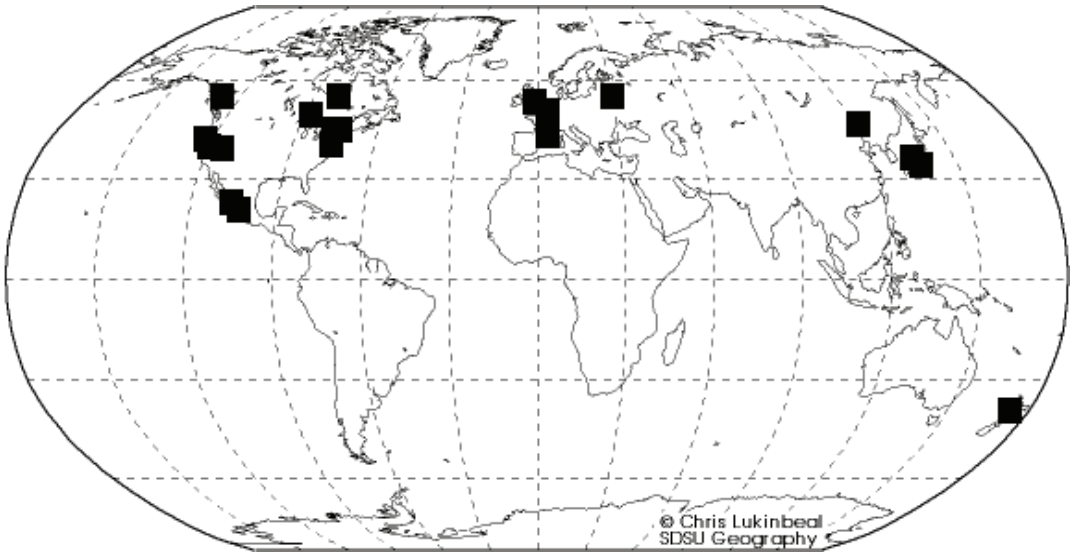


Figure 2.8: Distribution of CAIDA skitter monitors over the world

2.3.2 CAIDA Skitter

The Cooperative Association for Internet Data Analysis (CAIDA) provides tools and analyses promoting the engineering and maintenance of a robust, scalable global Internet infrastructure. CAIDA's Skitter tool deploys a method similar to traceroute to determine the IP path to a destination. Destinations are chosen from BGP tables and a database of Web servers. Skitter sends ICMP echo request packets, increments the TTL when sending them and registers the IP address of the replying routers. If a router does not respond to three subsequent ICMP request, the TTL is increased. When the desired destination is reached, skitter registers the round-trip-time (rtt). When TTL reaches 30, or "ICMP unreachable" reply has been received, Skitter stops probing the destination.

The CAIDA Skitter project has deployed around two dozens monitors worldwide (Figure 2.8). Each monitor performs traceroutes measurements to thousands of destinations every day. The traceroute paths of this thesis were obtained from the following skitter monitors: arin, b-root, cam, cdg-rssac, champagne, d-root, e-root, f-root, h-root, i-root, iad, ihug, k-root, lhr, m-root, mwest, neu, nrt, riesling, sjc, uoregon and yto. The arin monitor is located in Bethesda, MD, US. The b-root monitor is located in Marina del Rey, CA, US. The cam monitor is located in the University of Cambridge, Cambridge, UK. The cdg-rssac monitor is located in Paris, FR. The champagne monitor is located in the University of Illinois at Urbana-Champaign Urbana, IL, US. The d-root monitor is located in the University of Maryland, MD, US. The e-root monitor is located in NASA Moffett Field, CA, US. The f-root monitor is located in Palo Alto,

Figure 2.9: PlanetLab: Current distribution of 652 nodes over 318 sites

CA, US. The h-root monitor is located in US Army Research Lab, Aberdeen, MD, US. The i-root is located in Stockholm, SE. The iad monitor is located in Washington, DC, US. The ihug monitor is located in Auckland, NZ. The k-root and lhr monitors are located in London, UK. The m-root monitor is located in Tokyo, JP. The mwest monitor is located in San Jose, CA, US. The neu monitor is located in Northeastern University, Shenyang City, Liaonin Province, CN. The nrt monitor is located in Tokyo, JP. The riesling monitor is located in San Diego, CA, US. The sjc monitor is located in San Jose, CA, US. The uoregon monitor is located in University of Oregon, OR, US. The yto monitor is located in Ottawa, CA.

2.3.3 PlanetLab

PlanetLab is an open, worldwide distributed testbed that enables experiments under real-world conditions, and on a large scale (Figure 2.9). At the time of writing, there were more than 250 institutions participating in PlanetLab projects, including TU Delft. Currently, there are 652 nodes over 318 sites running on locations in USA, Asia, and Europe. Architecturally, the PlanetLab network is similar to RIPE: each node can serve as a source as well as a destination. From each of the PlanetLab sites, traceroutes can be performed to all the other PlanetLab sites.

Chapter 3

Hopcount and Degree Distributions in the Internet

3.1 Problem Statement

Chapter 2 presents what is known about measurement framework and metrics, and discusses the measurement's uncertainties and errors. The main goal of this chapter is to present the measurement of hopcount and node degree from different measurement projects (*i.e.* RIPE NCC, PlanetLab and CAIDA), and provide some more insight into possible hopcount and degree distributions.

The hopcount of a path is the number of nodes of that path. Traceroute is a useful tool to collect the hopcount on a source-destination path. The distribution of the hopcounts over all pairs of nodes allows us to understand the average distance on a network. The maximal hopcount over all the paths of nodes is referred to as the diameter of the network. Results of the measurements of the hopcount distribution are important to better understand the current topology and to propose a more efficient network infrastructure than the current Internet. These results will also help simulate more realistic network topologies. Van Mieghem *et al.* [102][103] have shown that the hopcount distribution of the Internet is well modeled by that of a random graph with uniformly or exponentially assigned link weights. The first focus of this chapter is to compare our measurement results with their work. We also studied the change on the hopcount distributions over time, which may indicate dynamic changes in the Internet.

The degree of a node is the total number of its neighbors connected. Although the degree of a vertex is a local quantity, we shall see that a degree distribution is useful in determining important global characteristics of the Internet structure. Degree distribution is therefore frequently used to simulate realistic network topologies. Faloutsos *et al.* [34] have shown that the degree distribution of the Internet obeys a power law, and found that Internet models before [34] failed to exhibit power laws. This result

had significant impact on network topology research [19][97][61][14][52]. Thus the second subject of this chapter is to compare our measurement results with the work of Faloutsos *et al.* [34].

In order to compare actual measurement results to the results mentioned above, some steps were performed. First of all, an authoritative project environment RIPE TTM has been chosen. Moreover, the routing paths with pathologies were removed, and the most dominant of source-destination paths, *i.e.* the paths occurring most frequently, were selected. We were therefore stimulated to conduct measurement-based analysis to answer the following questions:

1. "Which types of errors are observed in the current Internet routing based on the traceroute measurement?"

2. "How can the hopcount in the Internet be measured and how does our measurement compare to the mathematical models of Van Mieghem *et al.* [102][103]?"

3. "Is the Power law distribution in the node degree distribution in the Internet always observed?"

The errors occurred in Internet routing are poorly studied, the exception being Paxson's [76] analysis of the dynamic behavior of Internet routing in 1995. At that time, he analyzed $40,000$ end-to-end path measurements, made by repeating traceroute utility [94] among 37 Internet sites, analyzing some routing pathologies behaviors in detail. His data showed that around half of the paths were asymmetric. In 2000, H. Tangmunarunkit *et al* [40] investigated how both routing protocols and routing policies affect the paths on Internet, and concluded that about 20% of paths are enlarged for more than five hops. Those routing pathologies of our measurement results are classified and compared to Paxson's data in the next section. We found that the likelihood of encountering a major routing pathology in recent years is greater than in 1995.

3.2 Traceroute Routing Pathologies

We begin our study with the errors in traceroutes of the Internet. To determine the routing path information, TTM uses around the well-known traceroute program. The traceroute program is written by Van Jacobson to return the path, specified by a sequence of IP-address of routers along the path, from source to destination. In TTM, a traceroute between each source-destination pair is done approximately 10 times an hour. In the period 1998-2001, the total number of boxes was about 40. In the Internet there is no guarantee that IP packets will follow a same path from a source to a destination. Changes, *e.g.* link-failures, policy-updates or load balancing, lead to the changes in traceroute. For these 40 test boxes (which can be sources and destinations), $1,329,019$ different routes have been obtained ($1,329,019$ different routeIDs, see Figure 2.5) with a total of $24,181,803$ routes (the sum of the raws in count column for all source-destination paths). After removing some test boxes whose paths often met with

errors, 31 out of 40 test boxes were chosen for this study, corresponding to $\binom{31}{2} = 465$ most dominant paths. In order to collect the information from both senders and receivers, and for easy access from the same place as the data files, there is a central point in RIPE that collects all the traceroutes from each source-destination pairs. In order to investigate the diagnostics of routing paths in the current Internet, in this chapter we ask these questions: which types of errors do we observe in the Internet, what is the cause of these errors. Pathologies perceived in the behavior of routing paths have been classified on the bases of nomenclature proposed by Paxson [76], and compared to his results. First, we analyze the most frequently occurring traceroutes records of each source-destination paths in details, and then we proceed until the 8^{th} most frequently occurring traceroutes records.

3.2.1 * (star)

A star * indicates that a packet did not come back to the source. This is due to the packet got lost or thrown away. There are several factors responsible for lost packets. Some gateways do not return the appropriate message requested by traceroute. Some firewalls use packet filters which block packets used by traceroute. Finally, packets may be lost as a result of network congestion. Of the 465 most frequently occurring traceroutes, 25 exhibited temporary routing loops, 5.38% of all the records.

3.2.2 Routing Loops

An IP that appears in the traceroute more than once is called a routing loop.

Those loops are classified as either, "persistent routing loops" if traceroutes show loops that were not resolved by the end of the traceroutes (*i.e.* after probing 30 hops), or "temporary routing loops" if traceroutes resolve loops within the routes. Persistent routing loops are usually caused by mis-configurations or chronic instability in routing tables and hence usually show entire loops. Transient loops are mainly due to the dynamic nature of networks, which causes routing changes. Such loops typically resolve as the routing protocol adapts to the network change and routing states converge. We discuss these two types next.

Persistent routing loops

A persistent routing loop is easy to detect in its traceroute. The following is an example of a persistent routing loop between a testbox located in Amsterdam, NL and a testbox located in Paris, France.

1 x.x.0.14
2 x.x.0.54
3 x.x.0.244
4 x.x.0.54
5 x.x.0.244
...

This persistent loop (between x.x.0.54 and x.x.0.244) did not be resolved after a short amount of time. The traceroute was lost till to the end. This indicates the connectivity deteriorating before a routing change, leading to an inconsistent state. Of the 465 most frequently occurring traceroutes, 1 path exhibited persistent routing loops.

Temporary routing loops

With temporary routing loops, the traceroute probe traveled beyond the loop and attain to the destination. Here is an example of a temporary routing loop between a testbox located in Leeds, UK and a testbox located in Geneva, CH.

1 x.x.70.193
2 x.x.201.98
3 x.x.201.128
4 x.x.71.194
5 x.x.201.128
...

At hop 3 and hop 5, the traceroute detects an IP-address x.x.201.128. The temporary routing loops are mainly caused by the asynchronous updates of the topology information. Of the 465 most frequently occurring traceroutes, 10 exhibited temporary routing loops, 2.15% of all the records.

3.2.3 Summary

Figure 3.1 summarizes the analyzed routing pathology and compares them to results obtained by Paxson in 1995.

Among these first 8 most frequently occurring paths, the probability of user visible pathologies in RIPE TTM is larger than that of Paxosn's, suggesting that the quality of traceroute measurements appears to have decreased over time at that part of the Internet. The main reason for this increment is the unknow IP address (*) in RIPE TTM. This rule increases the probability of the routing pathologies. These results suggest that some devices (*e.g.* Cisco routers) apparently treat router-destined ICMP with a lower priority than normal traffic, or more firewalls use packet filters which block packets used by traceroute.

	* (star)	Persistent loops	Temporary loops	Total paths records
Paxson	/	0.13-0.16%	0.055 - 0.078%	40000
RIPE TTM 1st frequently occurring routing	6.38%	0.2%	1.11%	461703
2nd	4.01%	0.65%	1.37%	422295
3rd	5.3%	1.29%	1.14%	294699
4th	3.86%	1.72%	0.95%	261942
5th	5.31%	1.29%	0.93%	204858
6th	6.44%	3.01%	0.19%	185218
7th	6.81%	2.37%	2%	162320
8th	7.02%	2.15%	1.8%	151346

Figure 3.1: Comparing results obtained by Paxson and RIPE TTM

3.3 Construction of Three IP level Maps

3.3.1 Constructing Union of Shortest Paths Based on RIPE

For each pair of measurement boxes, a large number of different paths have been identified in the database. Obviously, some of the traceroute records suffer from errors. For 4 test-boxes (out of 40) no record could be found in the database, and traceroute records for 5 test-boxes were all erroneous (temporary or persistent loops and unresponsive routers). These 9 test-boxes have therefore been excluded. In the traceroute data from the remaining 31 boxes, the most dominant path, *i.e.* the path occurring most frequently, of each pair boxes has been determined, resulting in a total of 465 most dominant paths. We ascertained further that 17% of those traceroutes suffer from errors mentioned above. The graph G_1 has been created by including every link belonging to each of the remaining 386 non-erroneous paths, resulting in a graph consisting of 1888 nodes and 2628 links. Here we must make one important remark: the traceroute utility returns the list of IP addresses of routers along the path from source to destination. One router can have several interfaces, with several different IP addresses. To determine which IP addresses belong to which router is a rather difficult task due to, amongst other reasons, security (port snooping). As a consequence of this, the graph G_1 represents the approximation of the Internet interface map, not of the Internet router map.

3.3.2 Constructing Union of Shortest Paths Based on PlanetLab and CAIDA

We have performed two sets of measurements. The first one is based on PlanetLab, while the second one is based on CAIDA. Our traceroutes experiments on the PlanetLab were executed on 10 November, 2004. At that moment, there were 445 PlanetLab nodes running on locations in USA, Asia and Europe. Note in some cases there are multiple nodes per PlanetLab site situated in the same location, we selected one node per PlanetLab site, resulting in a total of 79 nodes. We eliminated all incomplete traceroutes (*i.e.* erroneous traceroutes), and extracted the stable traceroutes. Based on the traceroutes collected from these 79 sides, the underlying IP level topology, consisting of 4226 nodes and 7171 links, has been created.

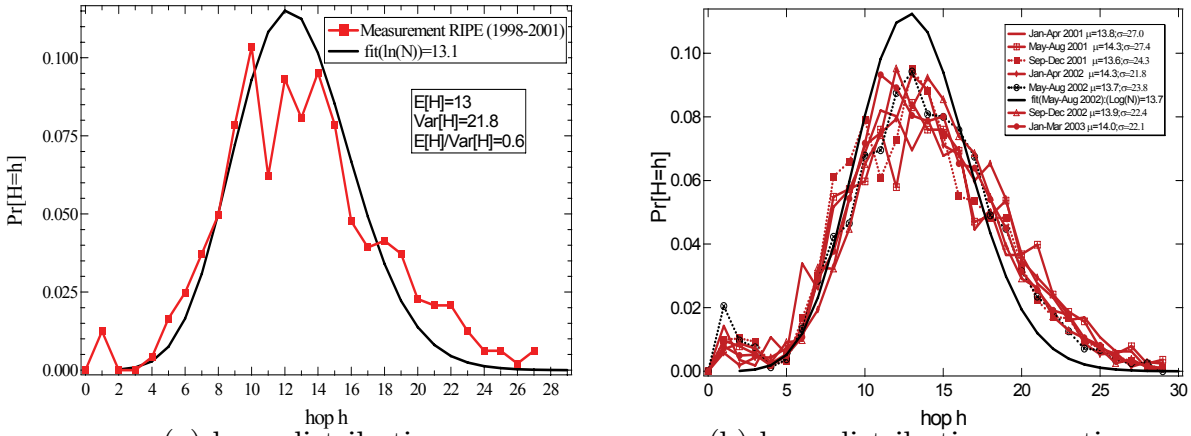
Our traceroutes experiments on the CAIDA skitter were executed on 12 May, 2004. At that moment, traceroutes paths were performed from 22 monitors to a large number of IP addresses (vary from a hundred to tens of thousands) in the world. There were total 976238 Internet traceroutes in the measurement. Similarly, we eliminated all incomplete traceroutes (*i.e.* erroneous traceroutes) and extracted the stable traceroutes. For example, 247106 out of total 427093 traceroutes are stable and complete in the monitor apan_jp, and those traceroutes had been further analyzed. This topology consisted of 112675 nodes and 140875 links.

By merging all traceroutes collect from RIPE, PlanetLab, and CAIDA as described above, three IP-level maps are obtained, from which the hopcounts and node degree distributions are computed.

3.4 Hopcount Distributions Based on RIPE, PlanetLab and CAIDA

First, the data from the RIPE network has been used. Intuitively, a much better approximation to the Internet graph would be a graph created as the union of k most frequently occurring paths. In our study, we have also considered the graph G_{10}, constructed as the union of the ten most frequently occurring paths, and we have discovered that the properties of G_{10} still resemble those of G_1. Therefore, due to the higher complexity of creating G_k, in the further analysis we will confine ourselves to G_1. Note that although the graph G_1 differs from the real underlying Internet graph G_{INT}, it appears to present an approximation to a part of G_{INT}. Moreover, nearly 85% of the total number of nodes is already spanned by the most dominant traceroutes from (any) 18 sources to all destinations. Including the most dominant traceroutes of a new source only adds 1.5% of new nodes to the topology.

In Figure 3.2.(a), the probability density function of the hopcount in the graph G_1 has been plotted. A peak with $h = 1$ is due to the fact that several test-boxes were



(a) hops distribution (b) hops distributions over time

Figure 3.2: The hopcount distributions based on RIPE NCC TTM data

located in Amsterdam and only 1-hop away from each other. The data of the hopcount has been fitted with a theoretical law 3.1 derived in [102][103] based on the random graph $G_p(N)$ (which consists of N nodes in which the links are chosen independently and with probability p) with uniformly (or exponentially) distributed link weights. The most important observation in the modeling [102][103] is that the shortest path tree in the random graph with uniformly (or equivalently exponentially) distributed link weights is independent of the link density p. This implies that even the complete graph with uniformly distributed link weights possesses precisely the same type of shortest path tree as any connected random graph with uniformly distributed link weights.

In a source-based shortest path tree, the link weight structure is more decisive than the underlying random topology. In [102][103] it has been shown that the shortest-path problem in the class of random graph $G_p(N)$ [10] with exponentially distributed link weights can be reformulated into a Markov discovery process with an associated uniform recursive tree [91]. A uniform recursive tree is defined by the following growth rule: given a uniform recursive tree with N nodes, a uniform recursive tree with $N+1$ nodes is deduced by attaching the $N+1$-th node uniformly (thus with probability $\frac{1}{N}$) to each of the N other nodes in the tree. If the number of nodes is N, the probability density function of the hopcount in the uniform recursive tree is shown [102][103] to precisely obey

$$\Pr[h_N = k] = \frac{(1+o(1))}{N} \sum_{k=1}^{N-1} c_{m+1} \frac{\ln^{k-m} N}{(k-m)!} \tag{3.1}$$

where c_k are the Taylor coefficients of $\Gamma(x)^{-1}$ listed in [1]. The above law can be approximated as a Poisson law,

$$\Pr[h_N = k] \approx \frac{(E[h_N])^k}{k!N} \tag{3.2}$$

which implies that $E[h_N] \approx Var[h_N] \approx \ln N$.

Figure 3.2.(a) illustrates that a fit with Equation 3.1 is reasonable.

In order to investigate the dynamic behavior of the hopcount in the current Internet, we studied the shorter term changes in the hopcount distributions (measurements repeated in every four months from $2001-2003$). Similarly, the graphs G_1 of each dataset with different periods have been constructed. In Figure 3.2.(b), the probability density functions of the hopcount in different graphs G_1 have been plotted. As verified from Figure 3.2.(b), there is virtually no large difference in the hopcount distributions over time. The results from this test suggest that there is no significant relation between the time of the measurement and measured hopcount, which suggests that the routing distance does not change significantly over time. This could be caused either by the fact that routers used relatively stable routing tables or by the fact that the updates of the routing tables do not affect the value of the hopcount in general. Moreover, Figure 3.2.(b) illustrates that a fit with Equation 3.1 is still reasonable for the graph G_1 constructed in the period May-August 2002. We observed the similar results for other graphs.

Next, the data from the PlanetLab network was used. By merging the traceroutes from each of 79 nodes to all the others, the topology consisting of 4226 nodes and 7171 links was produced. The hopcount distribution in this map has been computed and is depicted in Figure 3.3. Again, a high quality fit has been achieved for this figure.

Finally, the data from the CAIDA network has been used. The hopcount distribution in this map has been computed, and is depicted in Figure 3.4. Two experiments have been done. The subject of the first experiment is to present the distribution of the hopcount for different continents, while the second is to present the distribution of the hopcount for different sites. The results are depicted in Figure 3.4. As expected, the results for the Asian set of sites are slightly higher than for European and United States sets. This is caused by the greater geographical proximity of the sites in Europe.

The purpose of the second experiment is to present the measurement of hopcounts for different monitors. Possible differences between regions were therefore investigated, and the results are depicted in Figure 3.4.(b). The variation of the distributions is mainly due to that the monitors were placed in different parts of the network. As illustrated in Figure 3.4, high quality fits have been achieved with Equation 3.1 in these figures.

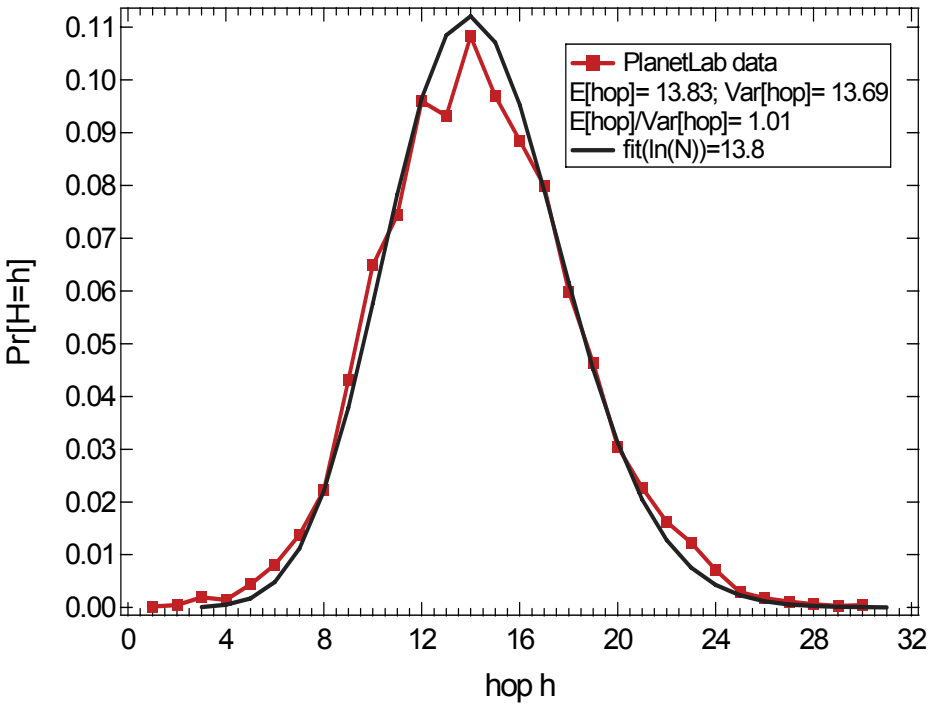


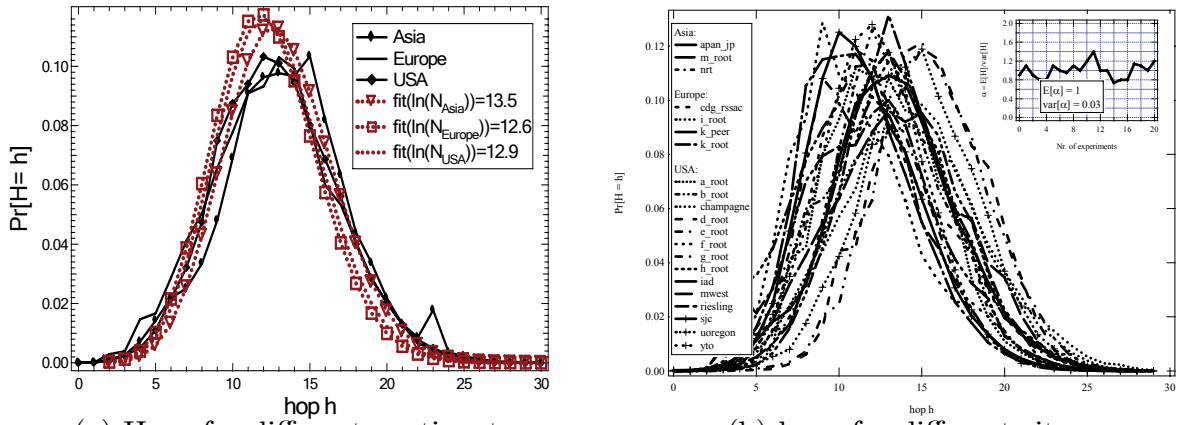
Figure 3.3: The hopcount distributions based on PlanetLab data

3.5 Node Degree Distributions Based on RIPE, PlanetLab and CAIDA

Node degree describes the connectivity characteristic of the topology. Networks with higher average node degree are "better-connected" on average and, consequently, are likely to be more robust. Next we use the maps obtained from RIPE, CAIDA, and PlanetLab to compute the node degree distributions of the Internet.

First, the data from RIPE was used. A map of the Internet was constructed from RIPE measurement data by assembling the most dominant non-erroneous traceroute paths from each of 31 test boxes to all the other boxes in the period $1998-2001$ (dataset 1), and 70 test boxes to all the other boxes in the period May-August 2002 (dataset 2). In this way, graph G_1 has been created, consisting of 1888 nodes and 2628 links, and 2574 nodes and 3922 links in two different periods respectively.

The results obtained were slightly different, as illustrated by Figure 3.5. We observe that the pdf of the node degrees in G_1 follows an exponentially decreasing function with a rate of -0.668 over nearly the entire range for dataset 1, and -0.61 for dataset 2. The figures also show the linear correlation coefficient r.



(a) Hops for different continents (b) hops for different sites

Figure 3.4: The hopcount distributions based on CAIDA Skitter data

The ratio of the average of the number of nodes with degree k, denoted by k , over the total number of nodes in the uniform recursive tree obeys for large N [91] which is, for large N, close to $Pr[degree = k]$, the probability that an arbitrary node has degree k. Hence, the decay rate of the pdf of the node degrees in the uniform recursive tree equals $-ln2 = 0.693$.

In summary, the pdf of the node degrees in G_1 (the union of the most dominant non-erroneous traceroutes) is similar to that in the uniform recursive tree. Hence, the close agreement points to the fact that the intersection of the trees rooted at a measurement box towards the other boxes is small, such that the graph G_1 is 'close' to a uniform recursive tree. This conclusion is verified as follows. Starting with the first union $T = T_1 \cup T_2$ of the trees T_1 and T_2 (where the tree T_i is a tree rooted at the box i towards the other boxes), we have computed the intersection (or overlap) $T_1 \cap T_2$. Subsequently, we have added another tree T_3 result in the union $T = T_1 \cup T_2 \cup T_3$ and again computed the intersection $(T_1 \cup T_2) \cap T_3$. This process was continued until all trees were taken into account. We found that the common nodes mostly augmented the degree by 1 and in very few cases by more than 1. It is likely that the overlap of trees would be larger (in terms of common nodes and links) if we had considered the router level map instead the interface map. The similarity in properties of the graph G_1 and the uniform recursive tree might be smaller in that case.

Second, the data from the PlanetLab network was used. By merging the traceroutes from each of 79 nodes to all the other, the topology consisting of 4226 nodes and 7171 links was produced. The node degree distribution in this map has been computed, and is presented in Figure 3.6. Again, a higher quality fit has been achieved on a log-lin than on a log-log scale, and the slope coefficient takes the value 0.48. This result seems to confirm our observation that when each source serves as the destination as well, the

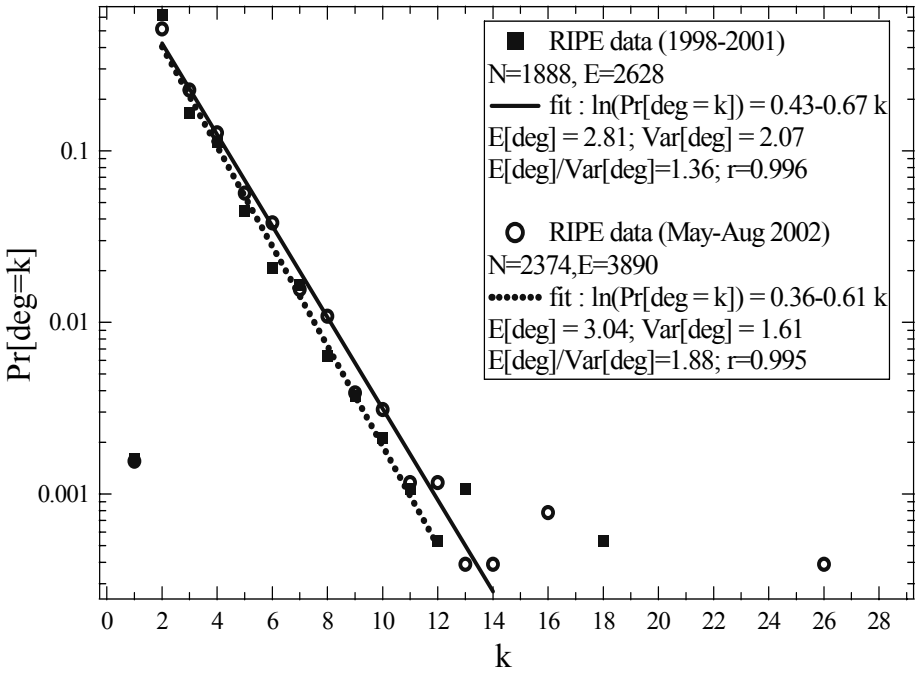


Figure 3.5: The node degree distributions based on RIPE NCC TTM data

node degree distribution observed seems to better follow an exponential function.

Finally, the data from the CAIDA network was used. As presented in Figure 3.7, a higher quality fit has been achieved on a log-log scale. The quality of the fit on the log-log scale implies a power-law. It is important to note the higher value of the slope coefficient, $\alpha = 2.5$. This conforms to the results of Faloutsos *et al.* [34], who have reported a power law for the degrees in the graph of the Internet.

The sample sizes S in the measurements of Figure 3.5 and Figure 3.6 were rather small (around 100 test-boxes) compared to that of Figure 3.7 (more than $20k$). Figure 3.5 and Figure 3.6 show different behavior than Figure 3.7. This result seems to suggest that G_1 of small group sizes in the core network (such as RIPE and PlanetLab) do not represent the graph of large group sizes (*e.g.*, CAIDA), thus may be not represent the graph of the Internet well. Our results have also shown that for large group sizes, the node degree distribution appears to obey a power-law, while for small group size, the node degree distribution appears better fitted with exponential distribution. Similar results have been reported by Milena [53]. Note that the global information of the Internet is unknown, while reliance upon a relatively small number of monitors to generate a graph of the Internet can introduce unwanted biases. For example, based upon an Internet topology collected from just twelve traceroute hosts by Pansiot and

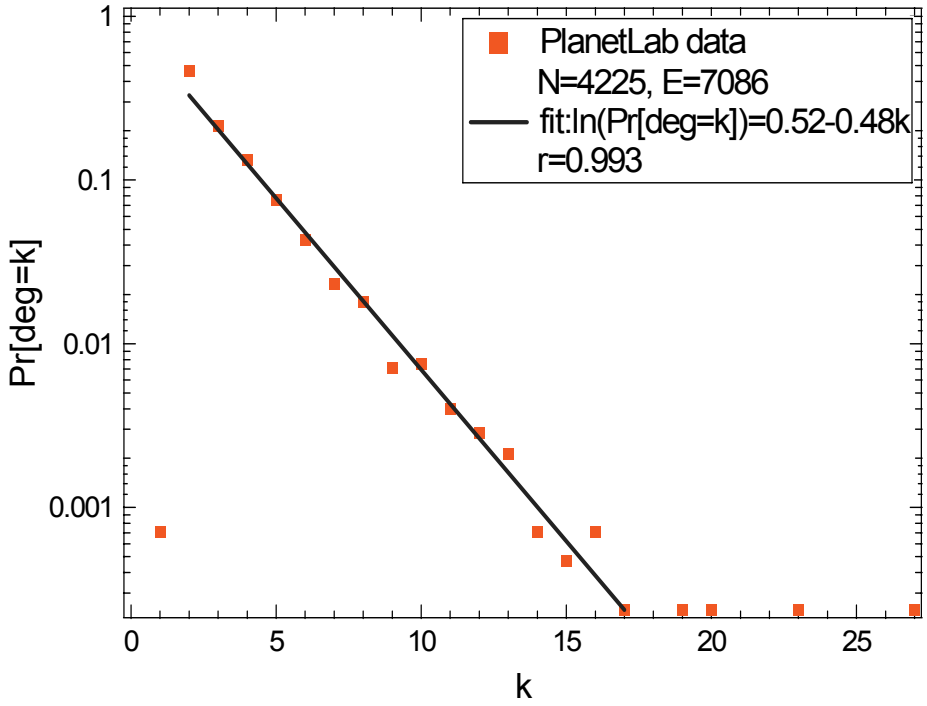


Figure 3.6: The node degree distribution based on PlanetLab data

Grad [73], Faloutsos *et al.* [34] found that the node degree distribution of the router map follows a power law. However, Lakhina *et al.* [57] showed that, in simulations of a network in which the degree distribution does not at all follow a power law, traceroutes conducted from a small number of monitors can tend to induce a subgraph in which the node degree distribution does follow a power law. Clauset and Moore [22] have later demonstrated analytically that such a phenomenon is to be expected for the specific case of the Erdos-Renyi random graphs [33]. All these results suggest the need for deeper investigation in how to accurately monitor the Internet [32].

3.6 Chapter Summary

The main goal of this chapter is to provide some more insight into possible hopcount and degree distributions, based on RIPE NCC, PlanetLab and CAIDA traceroute measurement data. Making a reliable hopcount measurement for certain region is not an easy task. One of the major problems is the routing pathologies. In the first place, therefore, traceroute routing pathologies must be analyzed and excluded for the further study.

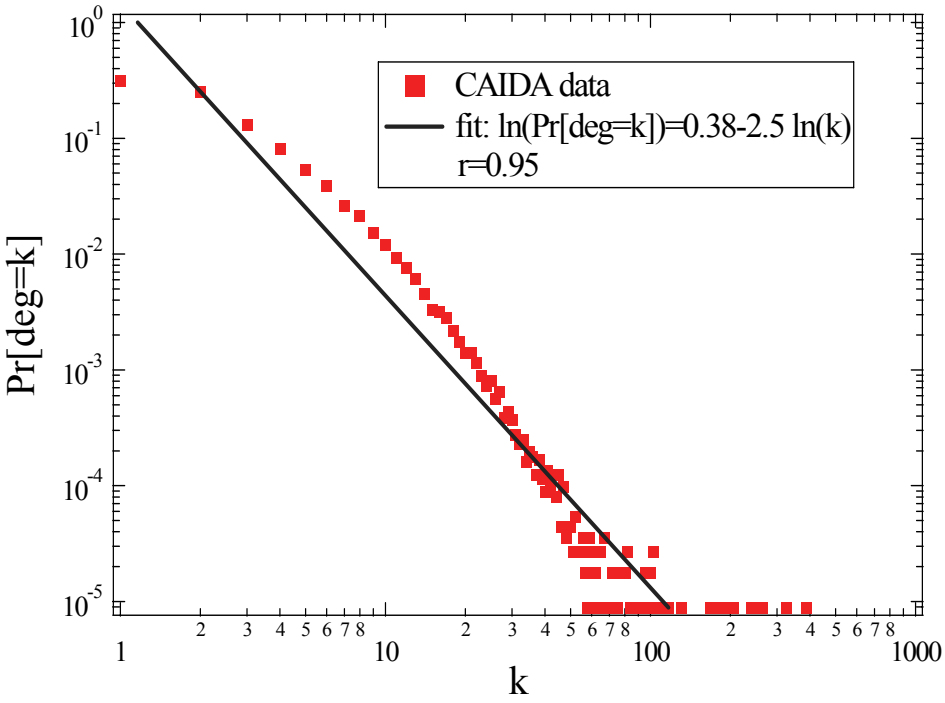


Figure 3.7: The node degree distributions based on CAIDA Skitter data

For this purpose, the traceroutes from RIPE NCC measured between 31 test-boxes on from 1998 to March 2001 were analyzed in depth. We examined characteristics of real networking, focusing mainly on understanding which types of errors users can observe in the current Internet routing, and understanding how the current routing algorithms affect the behavior of Internet paths.

The analysis of the pathologies in path routing has shown that compared with results obtained by Paxson in 1995, the value of errors increased. We may also conclude that the quality of traceroute measurements seems to have decreased over time. This is mainly due to that some devices (*e.g.* Cisco routers) apparently treat router-destined ICMP with a lower priority than normal traffic, or more firewalls use packet filters which block packets used by traceroute.

To calculate the accurate hopcount distribution and node degree distribution, graph G_1 has been constructed as the union of most frequently occurring paths in traceroute measurements for RIPE, and two IP level maps of the Internet have been constructed from PlanetLab and CAIDA measurement data, respectively. We compared our measurement results of hopcounts with the work of Van Mieghem *et al.* [102][103] based on the random graph with uniformly (or exponentially) distributed link weights. The experimental results showed that a resemblance to the pdf of the hopcount distribu-

tions in the Internet. Regarding the dynamic changes of the hopcount, the hopcount distributions did not exhibit large short-term changes. We further studied the node degree distributions from different datasets and our results show that for large group sizes (*i.e.* CAIDA), the node degree distribution appears to obey a power-law, while for small group size (*i.e.* RIPE and PlanetLab), the node degree distribution appears better fitted with exponential distribution. An accurate picture of the Internet is important, because researchers need models to examine the effects on traffic of new pricing mechanisms for ISPs or new routing protocols. Our results suggest the need for deeper investigation in how to accurately model the Internet.

Chapter 4

Reordering in the Internet

Chapter 3 investigates the hopcount and degree distributions in the Internet. In this chapter, we focus on the reordering in the Internet. Reordering, the out-of-order arrival of packets at the destination, is a common phenomenon in the Internet [75][8], and it frequently occurs on the latest, high-speed links. The major cause of reordering has been found to be parallelism in Internet components (switches) and links [8]. For example, due to load balancing in a router, the packets of a same stream may traverse different routers, where each packet experiences a different propagation delay and may thus arrive at the destination out-of-order. Reordering depends on the network load, although below a certain load very little reordering occurs. Reordering may also be caused by the configuration of the hardware (*i.e.*, multiple switches in a router) and software (i.e., class-based scheduling or priority queueing) in the routers.

The interest in analyzing end-to-end reordering is twofold. First, reordering impacts significantly on the performance of applications in the Internet. In a TCP connection, the reordering of three or more packet positions within a flow may cause fast retransmission and fast recovery multiple times, resulting in a reduced TCP window and, consequently, in a drop in link utilization and, hence, in less throughput for the application [59]. For delay-based real-time service in UDP (such as VoIP or video conference), the ability to restore order at the destination is likely to have finite limits. The deployment of a real-time service necessitates certain reordering constraints to be met. For example, in the case of VoIP, to maintain the high quality of voice, packets need to be received in order, and also within 150 milliseconds (ms). To verify whether these QoS requirements can be satisfied, knowledge about the reordering behavior in the Internet seems desirable. Second, these end-to-end reordering measurements may shed light on the underlying properties of the current topology and traffic patterns of the Internet.

Packet reordering is measured between 12 Internet testboxes of RIPE TTM (Test Traffic Measurement) [87] project. Our observations are based on packet level traces collected through the network. The main aim is the understanding of the nature of reordering. Our contribution is twofold. First, we describe the methodology used to

observe reordering behavior (Section 4.1). Second, we present our experiments (Section 4.2).

4.1 Problem Description and Definitions

A packet is classified as a reordered or out-of-order packet if it has a sequence number smaller than its predecessor. Specifically, let M streams, denoted as $(S_1, \ldots, S_M)$, be the total number of streams sent from node A to B. In each stream S_i consisting of K packets, we assign each packet j a sequence number a_j, which is a consecutive integer from 1 to K in the order of the packet emission. Thus we establish the source sequence as $(a_1, \ldots, a_K)$. Assume an output sequence $(b_1, \ldots, b_P)$ of stream S_i observed at the receiving node B, where $P \leq K$ is the total number of packets received out of the K packets sent. The amount $K - P$ is due to loss. The sequence is said to be in order if for each index k $(1 \leq k \leq P)$ holds $b_q < b_k$ $(0 < q < k)$, else the stream is said to arrive at the destination out-of-order, and the packet k is a reordered packet in the reordered stream. The total number of reordered packets in stream S_i is written as L_i. For example, for the sequence of an arrived reordered stream (1,2,3,5,4,7,6,8), there are 2 reordered packets (packet 4 and packet 6), which leads to $L = 2$. Note that in our thesis reordering does not correlate with loss (same as [68][4][79]). For example, a received stream (1,2,3,4,5,6,8) is considered to be in order.

We denote the reordered stream ratio by

$$R_{AB} = \frac{M_R}{M_a} \tag{4.1}$$

where M_a is the total number of received streams out of M streams sent and M_R is the total number of streams having at least one reordered packet. The reordering asymmetry is defined as the difference of two ratios $|R_{AB}\text{-}R_{BA}|$.

Let $U_n = \Pr[L_n > 0]$ denote the unconditional reordered stream percentage for the received stream n. And let C_n denote the probability that a stream $n + 1$ is reordered given that the previous stream n was reordered, defined by

$$C_n = \Pr[L_{n+1} > 0 | L_n > 0] \tag{4.2}$$

In order to predict whether a reordered packet will be useful in a receiver buffer with finite limit, for each reordered packet k $(1 \leq k \leq P)$, this chapter studies two more metrics: packet lag P_L and time lag T_L. Packet lag is proposed in [51] and refers to the number of packets k $(1 \leq k \leq P)$, with a sequence number greater than the reordered packet that has been received before the reordered packet itself. Thus,

$$P_L = \sum_{q=1}^{k-1} 1_{b_k < b_q} \tag{4.3}$$

where the indicator function 1_y is defined as 1 if the condition y is true, otherwise it is zero. For example, consider two packet sequences (2,1,3,4,5,6,7,8) and (2,3,4,5,6,7,8,1) which both consist of one reordered packet (packet 1), due to the different arrival positions of packet 1 in the two received sequences, then $P_L = 1$ for the former sequence, while $P_L = 7$ for the latter sequence. For a receiver with a finite buffer or a time constraint, recovering the latter sequence from reordering may be impossible.

Let t_k $(1 \leq k \leq P)$ be the 1-way delay of packet k. T_L is defined as the difference between the delay t_k of the reordered packet k and its expected delay $t_{k'}$ without reordering,

$$T_L = |t_k - t_{k'}| \tag{4.4}$$

In practice, $t_{k'}$ is replaced by $\min(t_1, .., t_P)$.

P_L is useful to evaluate the impact of reordering on TCP's performance since $P_L \geq 3$ would trigger the fast retransmit algorithms that halve the TCP sender's congestion window. We believe P_L is also a useful metric to study the impact of reordering events on UDP's performance. In addition to the P_L, T_L is a delay-based metric for more precise evaluation of the impact of reordered packets on the end hosts. For delay sensitive applications based on UDP, reordering can have a drastic effect on the application's performance. For example, in case of VoIP, to maintain the high quality of voice, packets need to be received in order and also before playback time. If a reordered packet arrives after its playback time has elapsed, that packet may be treated as lost.

4.2 Experimental Results

In the following we analyze the end-to-end packets reordering measurements performed in 12 "testboxes" of RIPE TTM project (see Section 2.3). We have analyzed the data collected between 12 test-boxes; where 3 hosts are located in the Netherlands, 2 in Great Britain, and 1 in Sweden, Slovakia, Belgium, Australia, USA, Denmark and Greece. 12 testboxes participated in the two experiments. First, between each sender-destination pair of measurement boxes, IP probe-streams of a back-to-back burst of 50 100-byte UDP packets, called probe-streams, were continuously transmitted with interarrival times of about 30 seconds, resulting in a total of about 360 probe-streams in 3 hours from 5 to 8 PM (Greenwich Mean Time) on October 16, 2003. Second, we repeated the same experiment with a burst of 100 UDP packets in 3 hours from 5 to 8 PM on October 17, 2003. In order to obtain snapshot of traffic patterns in the Internet, we limited each measurement to a total of 3 hours. The first experiment was labelled N_{50} and the second N_{100}. The difference between N_{50} and N_{100} may indicate how Internet packet dynamics change under two difference load situations. In a complete graph, ideally, the 12 test-boxes should consist of exactly 132 unidirectional links. In practice,

Received data	N_{50}	N_{100}
UDP streams	36762	32691
Reorder streams	20445	21649
UDP packets	1655120	2828834
Reordered UDP packets	101018	158413
Measurement duration	3 hours	3 hours

Table 4.1: Details of the packets used to measure the reordering on 104 paths

the experiment generally consisted of 104 unidirectional paths due to errors during the measurement.

To limit the influence of large packet loss, we only analyzed those streams which received at least 90% of all their total packets (*i.e.* each valid arrival stream has at least 45 UDP packets in N_{50} and 90 in N_{100}).

4.2.1 Reordered Probe-Stream Ratio R_{AB}

R_{AB} can provide insight into how often reordering occurred in the probe-streams. For each sender-destination pair we examined each arrival stream by checking its arrival sequence order. We calculated how many reordered probe-streams were received over 3 hours (there were approximately 360 probe-streams). Table 4.1 summarizes the total number of observed UDP streams and the packets in these streams on 104 paths over 3 hours. The results of our experiment (Table 4.1) indicate that reordering quite often occurs in the probe-streams. In N_{50}, about 56% of the probe-streams included at least one packet delivered out-of-order, while in N_{100} 66% did. Overall, 6% of all the received packets in N_{50} arrived reordered while N_{100} 5.6% were. It is interesting to note that this large fraction of reordering in streams is also reported by Bennett *et al.* [8] (over 90% with at least one reordering event). On the other hand, we found fewer probe-streams experiencing reordering compared to the number in [8]. This discrepancy may be caused by methodological differences between the studies: we used one-way UDP probe-stream, while the authors in [8] used TCP round-trip measurements.

We observed that a large fraction (>26%) of all UDP probe-stream suffered from reordering in all the experiment paths. In general, the probe-streams in N_{100} were more often reordered than those in N_{50}. For example, the average (over the 104 paths) is $E[R_{AB}] = 0.53$, where the standard deviation is $\sigma_{50} = 0.12$ in N_{50}. While $E[R_{AB}] = 0.65$ and $\sigma_{100} = 0.12$ in N_{100}. This is because higher traffic load probably contributes more to reordering.

We also observed that reordering varies greatly from testbox-to-testbox, for instance more than 70% of the streams transmitted from some testboxes in West Europe to two testboxes (a site in Australia, and another in Nottingham, Great Britain) in N_{50} arrived

out-of-order; far more than the 56% overall average, and the 80% in N_{100}. This is may have been caused by the heavy load at the links to these two testboxes.

4.2.2 Reordered Packet Lengths L

In order to quantify the extent of reordering, for each source-destination pair, we examined each arrival stream by checking its arrival sequence order and by calculating the reordered packet length L (the number of reordered packets).

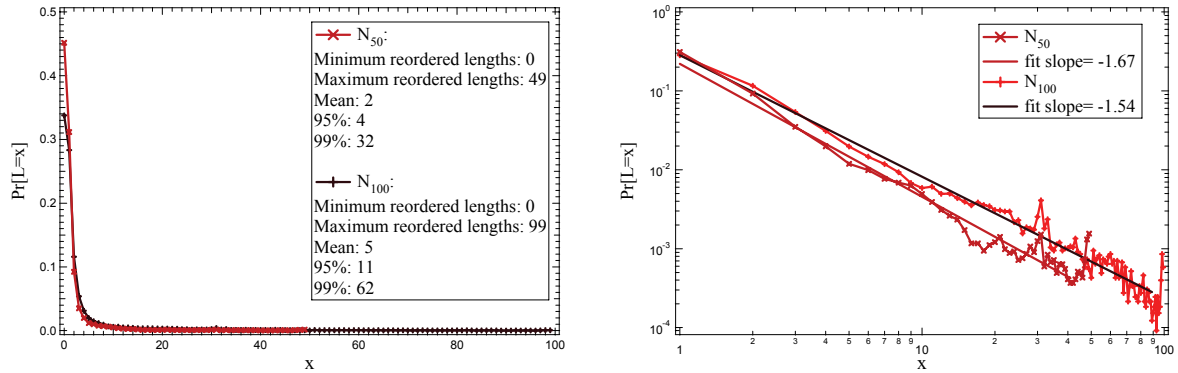


Figure 4.1: (a) The pdf of the reordered packet lengths L for the RIPE data sets. (b) The pdf of the reordered packet lengths L and the power law fit

Figure 4.1(a) plots a probability density function (pdf) of how many reordered packets are observed in N_{50} and in N_{100}. We found that the pdf of the reordered length has a relative heavy tail. Specifically, $L = 0$ for 44% of total probe-streams in N_{50}, $L = 1$ for 32% and $L = 2$ for another 10% of them. The maximum of L was 49 (about 0.15%). While $L = 0$ for 34% of N_{100}, and $L = 1$ for 28% and $L = 2$ for another 12% in N_{100}. This suggests that most individual reordered streams have a relatively small number of lengths. Fitting the probability density function of L on a log-log scale seems to indicate power law behavior for L. A power law is defined as $\Pr[L = x] \simeq C \cdot x^{-b}$, where the exponent b is the power law exponent and slope in a log-log plot. Figure 4.1(b) shows the exponent $b_{50} = 1.67$ in N_{50} and $b_{100} = 1.54$ in N_{100}, shown by dotted lines.

We found that each IP packet in a sequence had nearly the same probability of being reordered. This suggests that the cause of reordering acts upon a stream of IP packets almost as a Poisson process.

4.2.3 Packet lag P_L and Time lag T_L

In this section, we analyzed P_L and T_L on 104 unidirectional paths. To measure P_L, for each source-destination pair, we examined each arrival stream by checking its arrival sequence order. For each reordered packet in a reordered stream, we determined P_L by calculating how many packets with greater sequence numbers had been received before it.

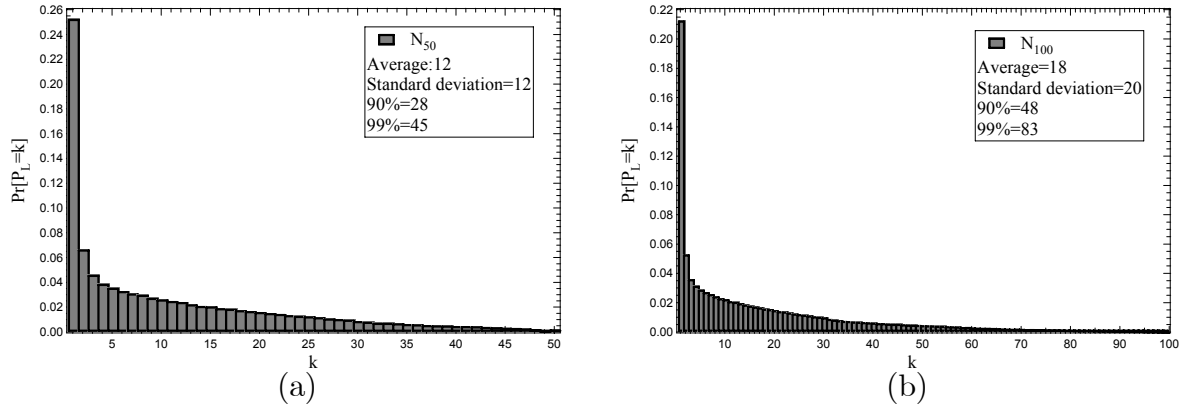


Figure 4.2: The pdf of packet lag P_L for 2 data sets

The resulting pdf of P_L (Fig 4.2) shows that only around 40% of reordered packets occurs within 1, 2 or 3 packets in N_{50}, while around 35% in N_{100}. Moreover, the long tails of the distributions (up to 49 in N_{50}, while 99 in N_{100}) certainly impact the UDP performance because the reordering adds a high recovery cost on the end host with finite buffer.

We then computed the time lags T_L of different reordered packets. For each source-destination, we examined each arrival stream by checking its arrival sequence order to determine the reordered packets. For each reordered packet, we determined T_L by calculating the difference between the 1-way delay and the minimal delay in its sequence $\min(t_1, ..., t_P)$. Each time lag T_L of a reordered packet was normalized by the minimal one-way delay of the packets in its sequence, thus

$$T = \frac{(t_i - \min(t_1, .., t_P))}{\min(t_1, .., t_P)} \tag{4.5}$$

A normalized time lag T around 0 means that T_L is very small compared to 1-way delay and a normalized T of 1 means that T_L is very comparable to 1-way delay.

Figure 4.3 shows the pdf of time lags normalized by the minimal one-way delay of its sequence. In our experiments 90% percentile of the normalized time lag was 5% of the minimal one-way delay, and 99% percentile of the normalized time lag was 40% of

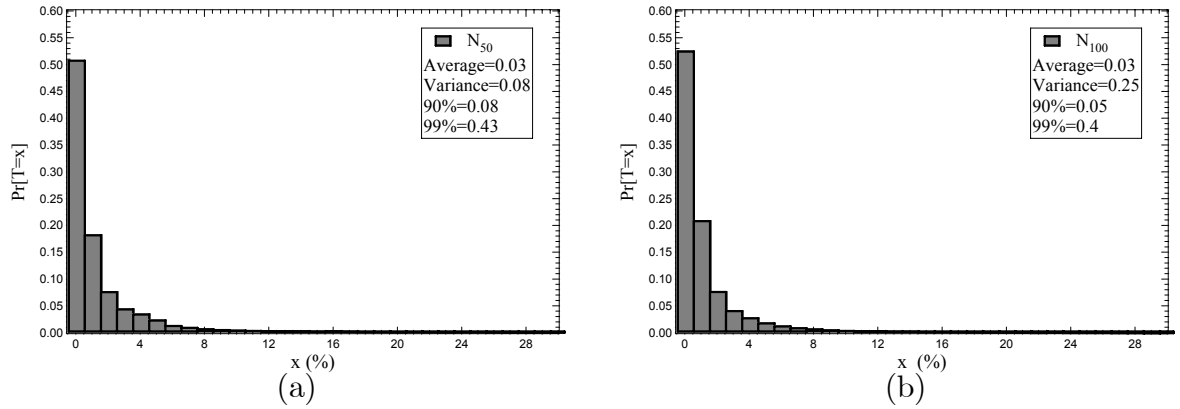


Figure 4.3: The pdf of normalized time lag T for 2 data sets

the minimal one-way delay. The Figures 4.3(a) and 4.3(b) indicate that most of the reordered time lag is very small, which suggests that packet reordering does not have a significant impact on the UDP delay since the reordering does not add large delay on the end hosts. However, the time lag can be also very large (up to 4 times the minimal one-way delay in N_{50} , while 17 times in N_{100}). Due to scale limitation, we did not show this large value in the figures.

4.2.4 Dependence of Reordered Probe-streams

For each source-destination pair, we calculated the unconditional reordered streams probability U. For each reordered stream, we examined whether the next stream was out-of-order or not to calculate the conditional stream probability C.

Figure 4.4 presents the measured values of the conditional reordering probability C and the unconditional reordering probability U in N_{50} and N_{100}: U is close to C for most paths. The weak dependence between two consecutive steams tells us that once a stream is reordered, the probability that the next stream is reordered does not seem to depend on whether the first was reordered or not. The effects that cause reordering seem to affect bursts at random, very similar to a Poisson process (which is memoryless).

These observations are expected: the interarrival time between streams is large (30 seconds). The measurement shows that this interarrival time is apparently long enough to treat the streams as independent.

4.2.5 Asymmetry of Reordered Probe-streams

Since unidirectional packet delay and traffic are highly asymmetric, it would be no surprise if reordering is asymmetric as well. For a UDP-based application, such as

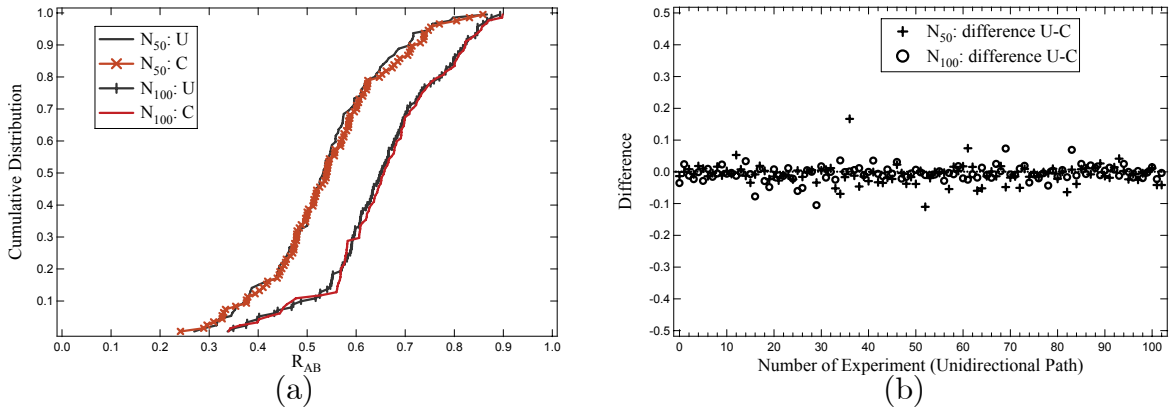


Figure 4.4: The difference between U and C for 2 data sets

VoIP, asymmetric reordering may result in a transaction in which one party has an acceptable quality of service while the other does not. In this section, we analyzed the asymmetry of reordered stream ratios R_{AB} and R_{BA}. We omitted pairs for which the probe-streams in one of the directions were missing or received less than 50% of all the streams sent (there are approximately 180 probe-streams), leaving data from a total of 39 pairs.

For the purpose of analysis, we defined the DAR, the degree of asymmetry of reordering, as:

$$DAR = \frac{|R_{AB} - R_{BA}|}{\min(R_{AB,}R_{BA})} \tag{4.6}$$

This function, DAR, plays a special role in the sense that it quantifies asymmetry, *i.e.*, a DAR around 0 suggests the difference of R_{AB} and R_{BA} is very small compared to $\min(R_{AB}, R_{BA})$. Figure 4.5 presents the plots of unidirectional packet reordering ratios of all traces in our measurements.

We observed that the asymmetry exists on all traces, but varies enormously from testbox-to-testbox. For example, depending on the measurement set, DAR ranges from just 0.001 up to 1.29. We noted that in N_{50}, 4 of the 38 traces have a DAR larger than 1, and 3 of the 4 traces consist of a testbox in Slovakia. We have also studied and observed a similar behavior for N_{100}. Beside this, 72% of the probe-streams in N_{50} experienced reordering on a path from Greece to Slovakia (where the average probe-packets delay (on this link) was 25.1 ms, whereas the standard deviation is 19.5 ms and the hopcount is 15), while 34% of the probe-streams experienced in the opposite direction (where the average probe-packets delay was 25.9 ms, the standard deviation 10.6 ms, and hopcount 17 or 18). We do not argue that this site-specific behavior reflects general Internet behavior, since [75] found that site-specific effects can completely change. However, we

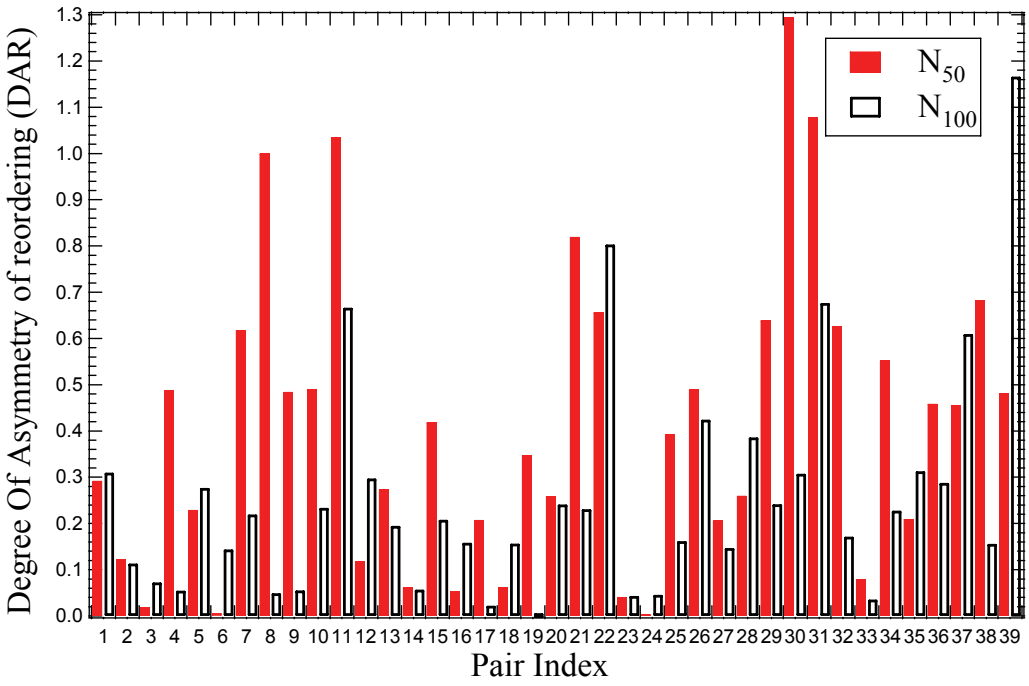


Figure 4.5: Degree of Asymmetry of Reordered streams in all 39 symmetric traces. The average (over the 38 traces) is $E[DAR] = 0.41$, $\sigma_{50} = 0.32$ in N_{50}, while $E[DAR] = 0.26$, $\sigma_{100} = 026$ in N_{100}.

suspect that the difference in stream reordering may be caused by the routing policies of the nodes on the path.

4.3 Chapter Summary

In this chapter, we have analyzed the measurements of the end-to-end packet reordering by tracing UDP packets between 12 testboxes in RIPE NCC. Our results lead to several observations:

- Reordering depends on the network load, although below a certain load very little reordering occurs [108]. For bursts of 50 100-byte UDP packets, there were about 56% of the probe-streams with at least one reordering event, and about 66% for bursts of 100 100-byte UDP packets.
- Most individual streams have a relatively small number of reordering events. For bursts of 50 100-byte UDP packets, there were about 14% of probe-streams with more than two reordering events, while about 26% for bursts of 100 100-byte UDP

packets. Furthermore, the heavy tails on Figure 4.1.(b) suggest that fitting the probability density function of reordered packet length L on a log-log scale seems to indicate power law behavior for L.

- Packet reordering has a significant impact on UDP performance since the reordering increases the high cost of recovery at the end host. On the other hand, packet reordering does not have a significant impact on UDP delay.
- The large interarrival time (30 seconds) between streams is apparently to be long enough to treat the streams as independent.
- Asymmetry of reordered streams ratios exists in all experiment pairs, but varies greatly from testbox to testbox.

Chapter 5

IPv6 Delay and Loss Performance Evolution

5.1 Problem Statement

This chapter focuses on the study of IPv6 delay and loss performance evolution. Although packet delay and loss are two important parameters of the Internet performance, to the best of our knowledge, the evolution of large-scale IPv6 [28] delay and loss performance has not been previously studied. Qualitative evaluation of IPv6 performance requires measurements collected over time, combined with information about delay, path, and tunnel discovery. Earlier studies mainly focused on IPv6 transition technologies [2] or identifying IPv6 network problems in a dual-stack world by using measurements from only a few days [20][105]. Compared to IPv4, IPv6 is still in its infancy and is rarely used by real-life applications. There is a lack of knowledge about the network performance of end-to-end IPv6 communication. In general it can be stated that the large scale deployment of applications will only be successful if the perceived quality of these applications is sufficiently high. Therefore, study of the large-scale IPv6 delay and loss performance evolution is important to understand the performance of the current IPv6 networks, and to provide high quality services for future Internet applications.

We investigate the IPv6 network performance in terms of delay and packet loss measured over the last two years. The data set for our study was obtained through measurements conducted by the RIPE NCC TTM project. At the time of writing, this data set contains active measurements for a set of about 26 test boxes supporting IPv6.

Our contribution is twofold. First, we present a measurement methodology to evaluate the IPv6 evolution performance by comparing IPv6 and IPv4 performance on a path-by-path basis (Section 5.3). Second, we investigate the different behavior of native IPv6 paths and IPv6 tunnel paths over time. We show that the bad performance of IPv6-in-IPv4 tunnels paths lead to worse performance of IPv6 compared to the IPv4

counterparts, while most native IPv6 show performance similar to that of their IPv4 counterparts (Section 5.4). On the other hand, there is little difference in loss performance between native paths and tunnel paths over time.

5.2 Background

5.2.1 The Transition Techniques From IPv4 to IPv6

IPv6 allocates 128 bits to represent an address, while IPv4 only allocates 32 bits. The number of different combinations therefore increases from 2^{32} to 2^{64} networks of 2^{64} addresses, which reduces the need for address translation. The use of Network Address Translators (NATs) sustains the explosion of end devices, but also greatly increases network complexity, which is a big barrier to the widespread introduction of point-to-point applications. IPv6 could enable every device to have its own IP address, alleviating the need for NATs. In addition, IPv6 also offers other advanced capabilities with respect to security, autoconfiguration and mobility.

Since a world-wide scale migration from IPv4 to IPv6 within a short period is unfeasible, three main transition techniques were invented to make the continuous transition from the current IPv4 Internet to IPv6 possible.

The first technique is the dual-stack network. This approach requires hosts and routers to implement both the IPv4 and IPv6 protocol using the same link layer. This enables networks to support both IPv4 and IPv6 services and applications at the same time during the transition period. Native IPv6 paths can be set up by enabling IPv6 on all routers and the links interconnecting them. At the present time, the dual-stack approach achieves a relatively good performance [2]. Although the dual-stack approach involves twice the complexity of single stack, it appears to be the natural choice in the long run.

The second technique relies on tunneling. Tunneling enables new IPv6 networking functions while still preserving the underlying IPv4 network as it is. For instance, when an IPv6 packet is leaving an IPv6 domain and entering an IPv4 domain, the packet is encapsulated in an IPv4 packet by a border router and transmitted through the network. When the packet reaches the other end of the IPv4 network, it is decapsulated at the border of the receiving IPv6 network. Tunnels can be statically or dynamically configured. 6to4 (*RFC* 3056 [17]) is a technique to transport IPv6 traffic over IPv4 networks without the need for automatic or configured tunneling. 6over4 (*RFC* 2529 [16]) is another technique that uses an existing IPv4 domain with multicast support to create a virtual link-layer for IPv6 hosts. Tunnels over Generic Routing Encapsulation (GRE) (*RFC* 2473 [25] and RFC 1701 [43]) have an extra encapsulation header to enable IPv6 traffic forwarding over an existing IPv4 infrastructure, with minimum changes.

This last technique uses a proxy and translation mechanism. Translation is necessary

in case neither tunneling nor native IPv6 are available, for example when an IPv6-only host wants to communicate with an IPv4-only host. An example of such technique is NAT-PT (Network Address Translator - Protocol Translator, *RFC* 2766 [100]), which performs address and protocol translation at the borders between non-homogeneous networks at the IP level. The drawback of using a translation mechanism is that there must be a mapping from each IPv4 address to an IPv6 address. NAT-PT therefore needs to use a pool of IPv4 addresses for assignment to IPv6 nodes on a dynamic basis as sessions are initiated across IPv4-IPv6 boundaries.

With different IPv6 transport mechanisms, IPv6 connectivity across the backbone can be set up with multiple segments managed independently. For example, an enterprise could decide to deploy a dual-stack network to connect to an ISP with a native IPv6, while connecting to another ISP with IPv6 over IPv4 tunnels. Since different organizations are at different stages in their transition to IPv6, we have a mix of native paths and tunnels as well as a mix of single- and dual-stack nodes today.

5.2.2 Current IPv6 Equipment Support

Since IPv6 routers with multiple very high-speed interfaces (such as Gigabit Ethernet, 10 Gigabit Ethernet) are not generally used, a good way to start an in-depth investigation of IPv6 performance is to know how IPv6 performance differs from IPv4 performance on a low-speed router (*e.g.* Cisco 3700 series). Depending on the router type and the implementation of the forwarding plane, IPv6 address lookups are performed by the software forwarding router (*i.e.* a device using its main CPU for basic and enhanced packet forwarding) and the hardware forwarding router (i.e. a device that has hardware assistance for basic and/or enhanced packet forwarding) [84].

Because the IPv6 lookup is more demanding (theoretically four times more demanding), there is a natural tendency to leverage hardware-based lookup engines (*e.g.* Cisco 12000 series and the Cisco Carrier Routing System) as much as possible. On the other hand, as the IPv6 header is considerably simplified, the classic IPv4 header contains 14 fields, whereas IPv6 only requires 8 fields, more efficient processing by routing nodes is possible.

Hardware-based IPv6 lookup designs generally lead to line-rate forwarding at all interface speeds for most packet sizes. Software processing of the IPv6 lookup (*e.g.* Cisco 7500 series router) takes more time than for IPv4 because more bits must be processed.

For example, when both IPv6 and IPv4 UDP packets (with different packet sizes) are forwarded through a Cisco-software-based forwarding platform (*e.g.* Cisco 3700 series), IPv6 and IPv4 forwarding performance of packets of large sizes (*i.e.* $\geq$ 128 bytes) are comparably good. However, when forwarding packets of small sizes (i.e. $\leq$ 128 bytes), IPv4 outperforms IPv6 by about 28% in terms of throughput [84].

If we repeat the experiment with a Cisco-hardware-based forwarding platform (*e.g.*

Cisco 12000 OC48), the forwarding performance at small packet sizes (*i.e.* $\leq$ 128 bytes) improves because of the hardware implementation. In this case, the IPv6 and IPv4 forwarding performance are comparable [84]. The other advantage of hardware forwarding is that IPv4 and IPv6 traffic will not compete for processor resources.

However, it is important to remember that the IPv4 infrastructure still remains the main source of revenue for ISPs and supports the most important services. Introducing IPv6 into these networks must not impact the current IPv4 network negatively.

5.3 Methodology

To undertake an active data traffic measurement study, we designed our methodology as follows. First, we gathered RIPE IPv6 data which includes traceroute, delay, and loss measurements among a list of IPv6 sites since 2003. (One-way delay and loss are defined in Section 2.1). Second, we also ran a tunnel discovery mechanism to distinguish between native IPv6 paths and tunnelled paths and studied their differences. Finally, based on our observations, we draw up a list of challenges for measurement and analysis in IPv6 networks.

5.3.1 Experimental Setup Review: RIPE TTM

Our data set was provided by the RIPE NCC TTM project. At the time of writing, the TTM infrastructure (which is solely in the core network) consists of approximately 26 IPv6 measurement boxes scattered over Europe, Asia and USA. Between each path of measurement boxes, both IPv6 and IPv4 UDP packets with a fixed payload (100 bytes) were continuously transmitted with inter-arrival times of about 30 seconds, and the hopcount between two measurement boxes measured every 6 minutes using traceroute. For simplicity, we assumed that those packets between two traceroute measurements used the same traceroute path. IPv6 paths were similarly monitored using traceroute6.

Next, a Maximum Transmission Unit (MTU) detection algorithm, written by Lorenzo Colitti [24], was run once per hour. In addition to the traceroute measurements, we used a tunnel discovery tool to identify different tunnels on these IPv6 paths. The tunnel discovery tool detects an IPv6 tunnel by measuring the MTU size (normally 1500 bytes) over an entire path. If a path contains a tunnel, the MTU on that path will usually be lower than 1500 since extra headers are added to the packet. However, this method is not perfect as not all links have an MTU 1500. Another problem is that the tunnel discovery tool cannot detect more than one tunnel. For instance, if there is an IPv6-in-IPv4 tunnel (MTU 1480) followed by a GRE tunnel (MTU 1472), the tunnel discovery tool is only able to detect the second tunnel (which has the lowest MTU). In any case, a returned value of "1500" indicates a native path, anything lower probably means that the path contains at least one tunnel. The MTU value of a tunnel

depends on the specific tunnel type: 1480 for IPv6-in-IPv4 tunnels; 1476 and 1472 for GRE tunnels; and 1280 for BSD tunnels. The IPv6 specifications define several types of IPv6-in-IPv4 tunnels, including configured tunnels and automatic tunnels (*RFC* 2893 [38]), 6to4 (*RFC* 3056 [17]), ISATAP (*RFC* 4214 [99]), and Teredo (*RFC* 4380 [45]).

5.3.2 Research Challenges

Before presenting the analysis, we formulate the two research challenges.

The *first* is in analyzing the large measurement database. We have analyzed more than 400 GB of zipped data collected over 2 years (from 1 Oct. 2003 to 31 Oct. 2005). Figure 5.1 shows the numbers of active IPv6 and IPv4 testboxes in the TTM infrastructure over 2 years. Note that not all boxes are active all the time due to reasons such as system updates or failures. The number of IPv4 paths is much larger than the number of IPv6 counterparts. Figure 5.1 also shows that the numbers of active IPv6 testboxes and paths have been steadily increasing over time. On Oct. 1, 2003, there were 15 active IPv6 testboxes with 210 active IPv6 source-destination paths; by the end of Oct. 2005, these numbers had increased to 29 and 811, respectively. For fair comparison, only those testboxes supporting both IPv4 and IPv6 traffic were selected in our study.

The *second* challenge is that the high dynamic evolution of IPv6 tunnels adds analysis complexity. Some IPv6 tunnel paths changed to native paths, and vice versa. Figure 5.2 shows the numbers of active native and tunnel paths over the last 2 years. Figure 5.2 shows that in Oct. 2003 about 61% of the total IPv6 paths were native paths, while by the end of Oct. 2005, this number had increased to 86%. We observe that there were 328 IPv6 tunnel paths before 30 Aug. 2005, while about 31% of those paths changed to IPv6 native paths after that, since several ISPs upgraded their routers to dual-stack for better performance.

Although IPv6 will replace IPv4 in the future, it is expected that IPv4 and IPv6 hosts will coexist for a substantial time during the steady migration from IPv4 to IPv6. It is important to understand how to measure and test IPv6 native/tunneling performance. To qualitatively evaluate the current IPv6 infrastructure, we use IPv4 performance as a comparison base, and compare the IPv6 delay and loss performance and the corresponding IPv4 performance in a path-to-path basis.

5.3.3 Presentation of the Data

Since minimal, average, and maximal IPv6 and IPv4 delay are sensitive to the clock error, to minimize this error, for each Src-Dst path, 2.5 percentile, median and 97.5 percentile delay are shown instead. For each Src-Dst path i, we first made the delay histogram distribution over a time interval (*e.g.* one day). Next, we computed the 2.5 percentile $D_{2.5}(i)$, median $D_{50}(i)$ and 97.5 percentile $D_{97.5}(i)$ delay values for that

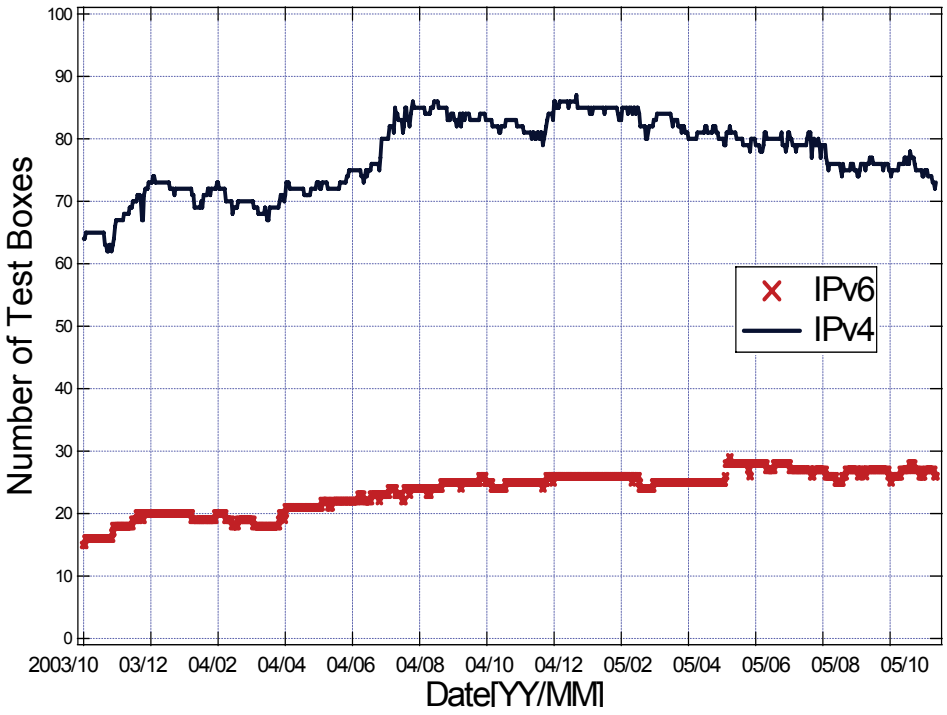


Figure 5.1: The number of active testboxes over time

path. We repeated this experiment for all the paths. When all paths were computed, we presented the average values of the 2.5 percentile, median and 97.5 percentile values of all paths. For example, $\frac{1}{n}\sum_{i=1}^{n} D_{2.5}(i)$ is shown in the following graphs of 2.5 percentile values, where n is the number of paths in that time interval.

5.4 Delay and Loss Performance

5.4.1 Evolution of Delay Performance of all TTM Paths over Two Years

First, we compare the general IPv6 delay and loss performance with their corresponding IPv4 performance over time, then we study and discuss the native IPv6 and IPv6 tunneled paths performance, separately. Figure 5.3.(a), (b) and (c) show the average 2.5 percentile, median and 97.5 percentile over two years with one day intervals, and Figure 5.3.(d) shows the loss comparison between IPv6 and IPv4.

From Figure 5.3.(a), (b) and (c), it can be estimated that on average IPv6 has about 61% higher 2.5 percentile delay than the IPv4 counterparts, and about 64% and 157%

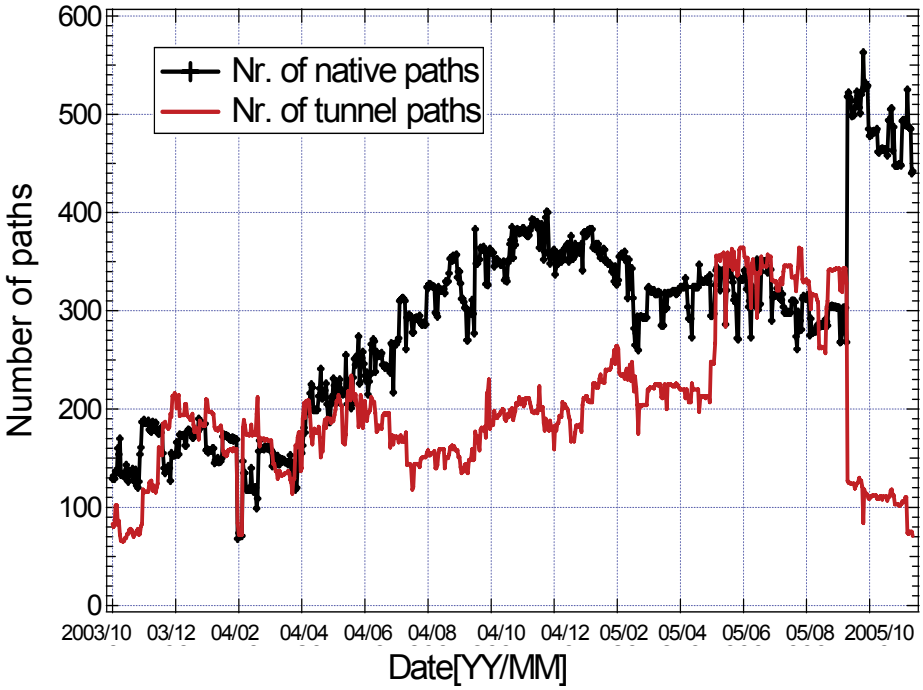
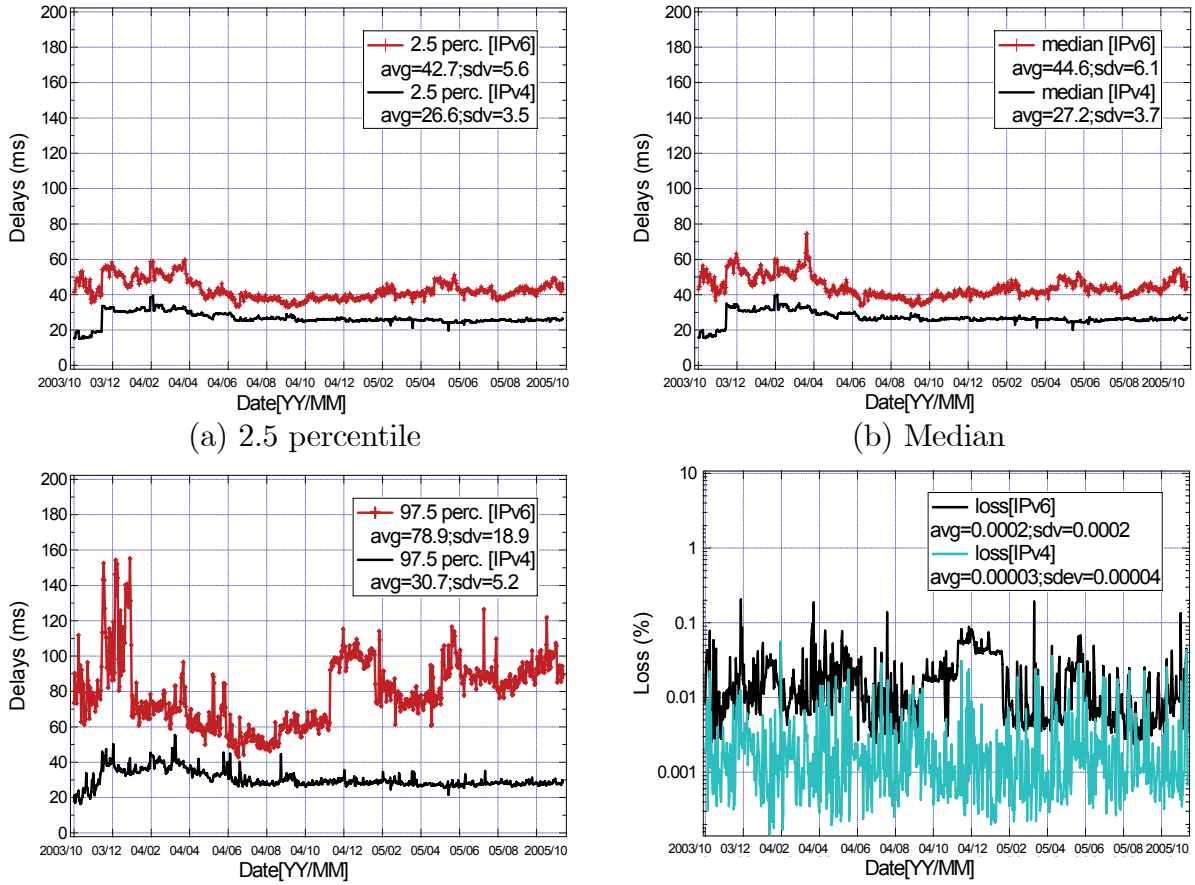


Figure 5.2: The number of active IPv6 native and tunnel paths over time

for the cases of median and 97.5 percentile delays respectively. Hence, both average and variation of the delay are worse for IPv6. Figure 5.3 shows that since late 2003, IPv6 has a larger average delay than the IPv4 counterpart. Over time, the latter has been steadily and slightly decreasing, however, the former shows relatively large variation, which has not been affected over the time.

Figure 5.3.(d) shows that the packet loss of all paths over the two years between IPv4 and IPv6 are small (less than 0.02%), and do not change much over time. On the other hand, our results show that over the years, IPv6 loss is about 1 order of magnitude larger than IPv4 loss. Due to the development and enormous diversity of the Internet, different average packet loss rates are reported in different studies: Boralla *et al.* [11] show the packet loss rates between 0.36% and 3.54% based on the study of speech data transmission in 1999. While Wang *et al.* [105] show average packet loss rates of 3.09% and 0.76% for the IPv6 and the IPv4 connections in 2005 respectively. Our results are considerably smaller than theirs mainly because of the experimental setup: we sent the probe packets (100 bytes UDP packets) about every 30 seconds in the TTM infrastructure, which is solely in the core network.

Based on the above measurements, we conclude that even though new testboxes were added into the measurement testbed in two years, the IPv6 and IPv4 delay and



(a) 2.5 percentile (b) Median

(c) 97.5 percentile (d) Loss performance

Figure 5.3: Performance of IPv6 paths and the IPv4 counterparts over time

loss did not change significantly, and IPv4 outperformed IPv6 in term of delay and loss during the whole period.

However, given the large percentage of tunneled IPv6 paths in current Internet (see Figure 5.2), it is not enough to know the aggregated measurements. The following three questions therefore still remain:

1. How differently do IPv6 tunnels paths behave compared to their IPv4 counterparts over time?

2. How differently do native IPv6 paths behave compared to their IPv4 counterparts over time?

3. Can the difference here explain the reason why IPv6 is outperformed?

Delay performance of native IPv6 paths

Figure 5.4 shows the 2.5 percentile (a), median (b) and 97.5 percentile (c) delay performance of the IPv6 native paths versus their corresponding IPv4 counterparts. The results in Figure 5.4.(a) and Figure 5.4.(b) suggest that the 2.5 percentile and median delay of those native IPv6 paths are slightly worse (23% and 27% higher) than those corresponding IPv4 paths. Similar to [2], our results show that native IPv6 can achieve a relatively good performance.

However, the results in Figure 5.4.(c) indicate that the 97.5 percentile delays of those native IPv6 paths are much worse (157% higher) than those for corresponding IPv4 paths. As argued in Section 5.4.4, IPv6 packets may have a lower priority than IPv4 ones, and thus have a longer delay in the processing time in the rush hours. These results indicate that in the most cases, IPv6 native paths have similar performance as IPv4 paths, except for a few with worse performance.

Figure 5.4 also shows that before 30 Aug. 2005, both IPv4 and IPv6 2.5 percentile and median delays slightly decreased over time, but increased by about 120% and 73% after that date. This difference was mainly caused by a change to about 30% long-distance IPv6 tunnel paths which were switched to native ones (see jump in Figure 5.4.(a)(b)) at that date. This increased the number of native paths, and also added extra average delay of these native paths. The observations also hold for the IPv4 counterparts since more long-distance paths were suddenly included.

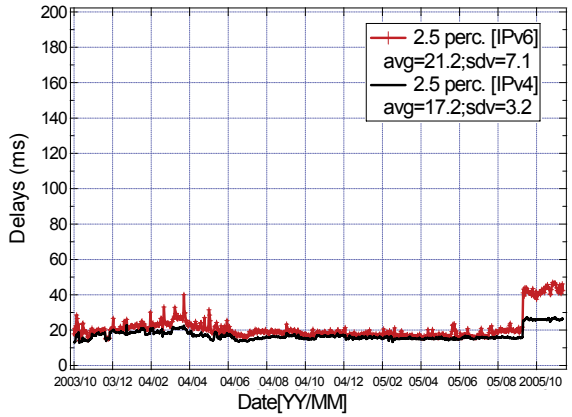
It is generally expected that a router's IPv6 forwarding performance will be similar to its IPv4 forwarding performance and close to the line rate of the tested interface.

IPv6 tunnels delay performance

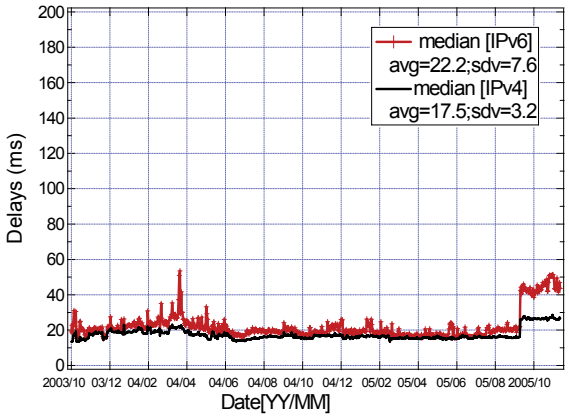
Figure 5.5 shows that IPv6 tunnel paths have a larger average delay than their IPv4 counterparts over time. It is also expected that when considering tunneling transition mechanisms, IPv6 traffic performance will degrade, since IPv6 packets have to be encapsulated in IPv4 packets and have an additional overhead. Our results show the evidence for this. On average, IPv6 has 2.5 percentile delay about 83% higher than their IPv4 counterparts (Figure 5.5.(a)), while about 87% and 165% for the cases of median and 97.5 percentile delays (Figure 5.5.(b)(c)) respectively. Compared to their corresponding IPv4 counterparts, IPv6 native paths perform a little worse, while IPv6 tunnel paths perform much worse. The difference here shows that tunnels degrade the network delay performance. Tunnels probably explain why IPv6 is outperformed by IPv4. Similar results have been reported in [24][20].

Broadly speaking, both IPv6 tunnels and IPv4 counterparts curves in Figure 5.5.(a)(b) have slightly decreased over time. However, the former shows a relatively larger variation.

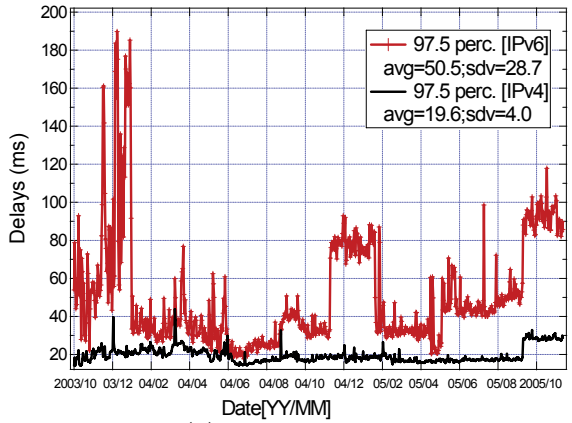
Figure 5.5.(d) shows loss performance for native IPv6 paths and tunnel paths over



(a) 2.5 percentile

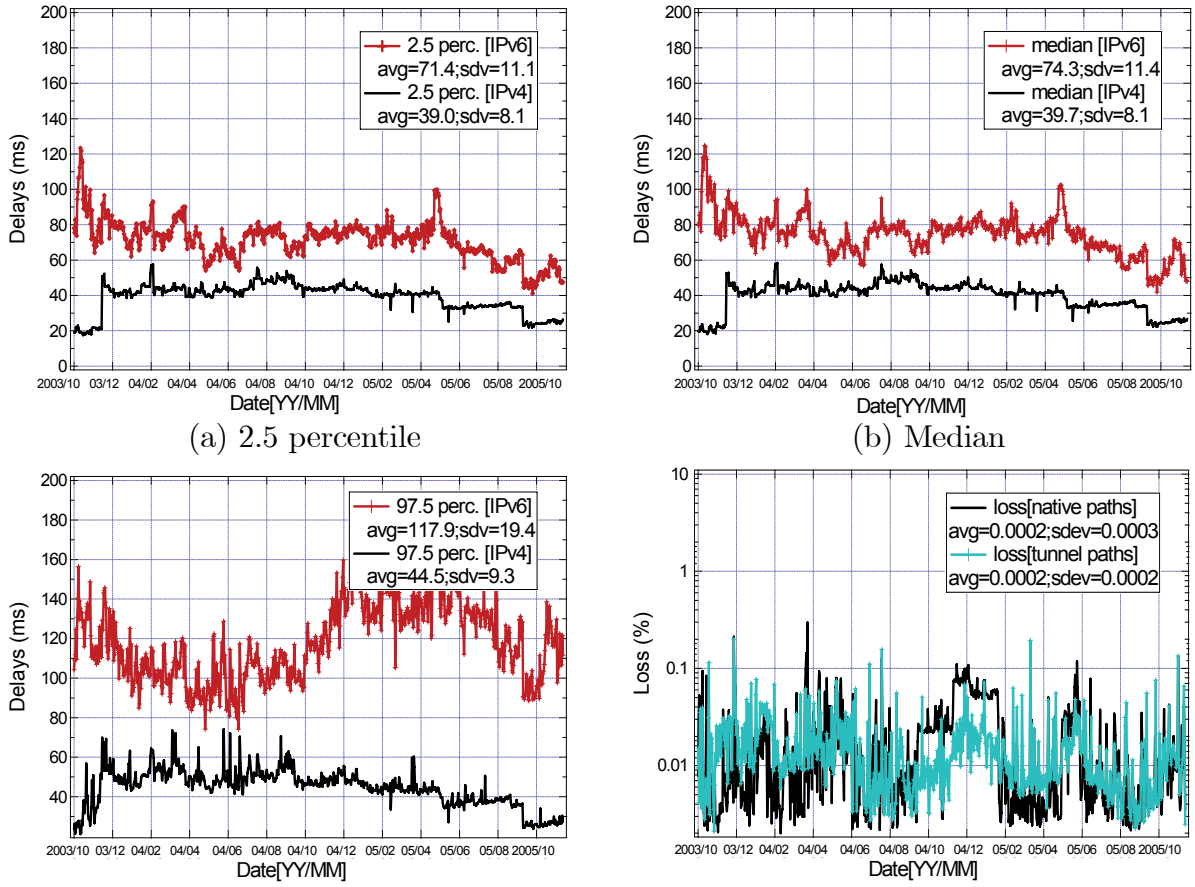


(b) Median



(c) 97.5 percentile

Figure 5.4: Native IPv6 paths and their IPv4 counterparts over time



(a) 2.5 percentile

(b) Median

(c) 97.5 percentile

(d) Loss performance

Figure 5.5: IPv6 tunnel paths and their IPv4 counterpats performance over time

time. We observe that the loss for both native IPv6 paths and IPv6 tunnel paths are roughly equal and very small (0.02%), and do not change much over time. While it may seem likely that since tunneling degrades the delay performance, it will also result in higher packet loss. However, our measurement results do not imply any strong correlation between tunneling/native and packet loss rate. This may be due to the available high bandwidths in tunneling and native.

Nevertheless, the delay performance difference compared to IPv4 is larger for IPv6 tunnel paths than for IPv6 native paths. This can be partly explained by the fact that almost all IPv6 tunnels are software-based and as described earlier. Software-based implementations incur more delay than hardware-based. Another reason could be the less optimal routing of IPv6 paths and their tunnels than IPv4. This may have a larger impact.

However, the precise reasons for the worse delay performance of IPv6 tunneling are unclear, since the management information of IPv6-in-IPv4 tunnels are unknown. These tunnels have in common that after configuration they behave like point-to-point links and only appear as one hop in traceroute measurements. We will return to this point in section 5.4.4 when we investigate single paths.

5.4.2 Delay Trends of Two Source-Destination Paths over Two Years

Figure 5.6 shows the delay trend of a typical tunnel path and that of its IPv4 counterpart (from a testbox located in Amsterdam to a testbox located in Dublin) over two years. Each point in the graph is the average delay measured over 6 hour intervals (4 sampling points per day). Figure 5.6 shows that IPv6 packets via a tunnel have a larger delay (mean IPv6 median delay is 31.72 ms, while that of IPv4 delay is 9.76 ms) as well as a much larger variance. Figure 5.6 also indicates that after two years evolution, the 2.5 percentile and median IPv6 delays of this path approach that of the IPv4 delays, which suggests that the IPv6 performance was improving. However, the 97.5 percentile IPv6 delays are much larger than the IPv4 counterparts. Another interesting observation is that there are some high peaks in the IPv6 delay, which suggest very poor delay performance. One such peak lasted from 4 Oct. 2005 to 21 Oct. 2005. Investigation of the traceroutes from those days reveals serious routing and/or link failures. This may also be caused by mis-configurations or chronic instability in routing tables, or because the routing had been manually changed.

Figure 5.7 shows a delay trace of a typical native IPv6 path and that of its corresponding IPv4 counterpart (both testboxes located in Amsterdam) over two years. The key observation is that the native IPv6 path displays similar behavior to that of an IPv6 tunnel path: the 2.5 percentile and median IPv6 delays approach that of IPv4 delays, but still display a much higher 97.5 percentile delay. This result suggests that

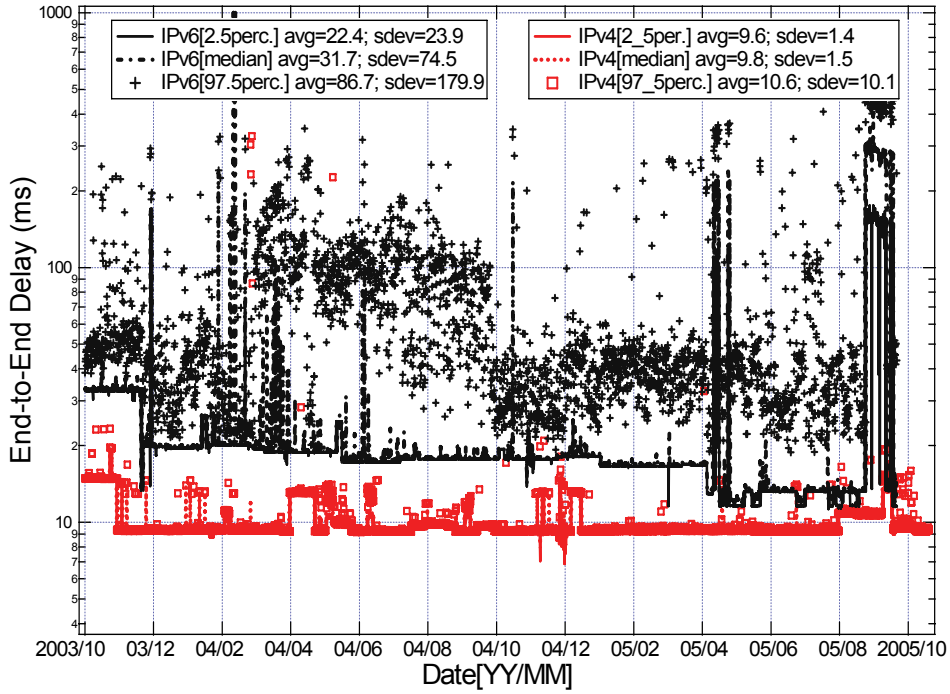


Figure 5.6: Delay trends of a tunnel pair over 2 years

in the worst delay cases (97.5 percentile), IPv6 tunnel and native paths perform much worse than IPv4, and the high peaks (large delay) in the IPv6 performance were not all caused by tunneling.

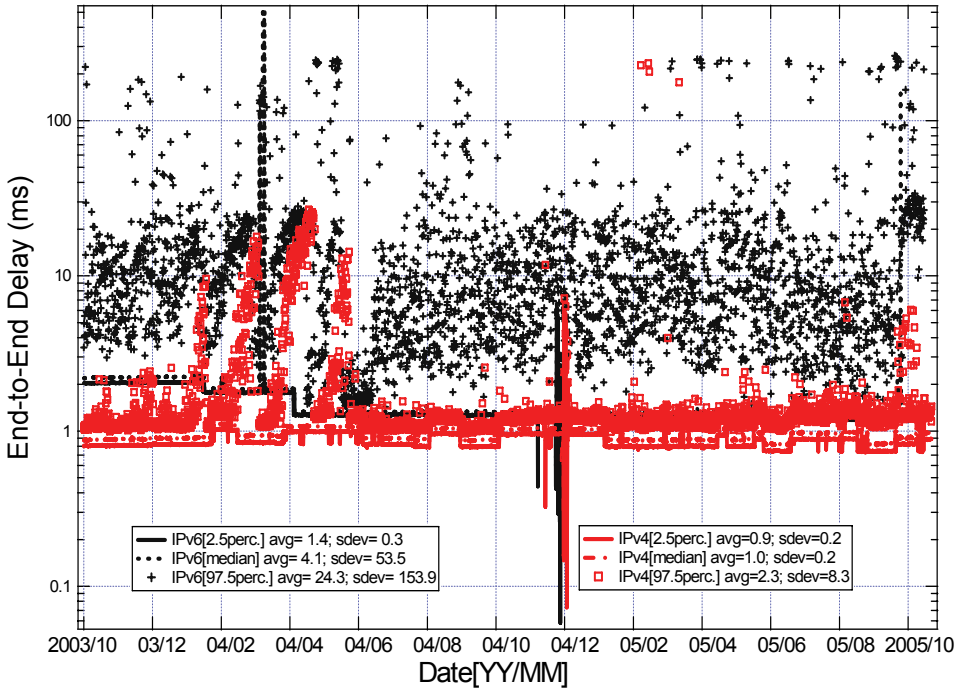
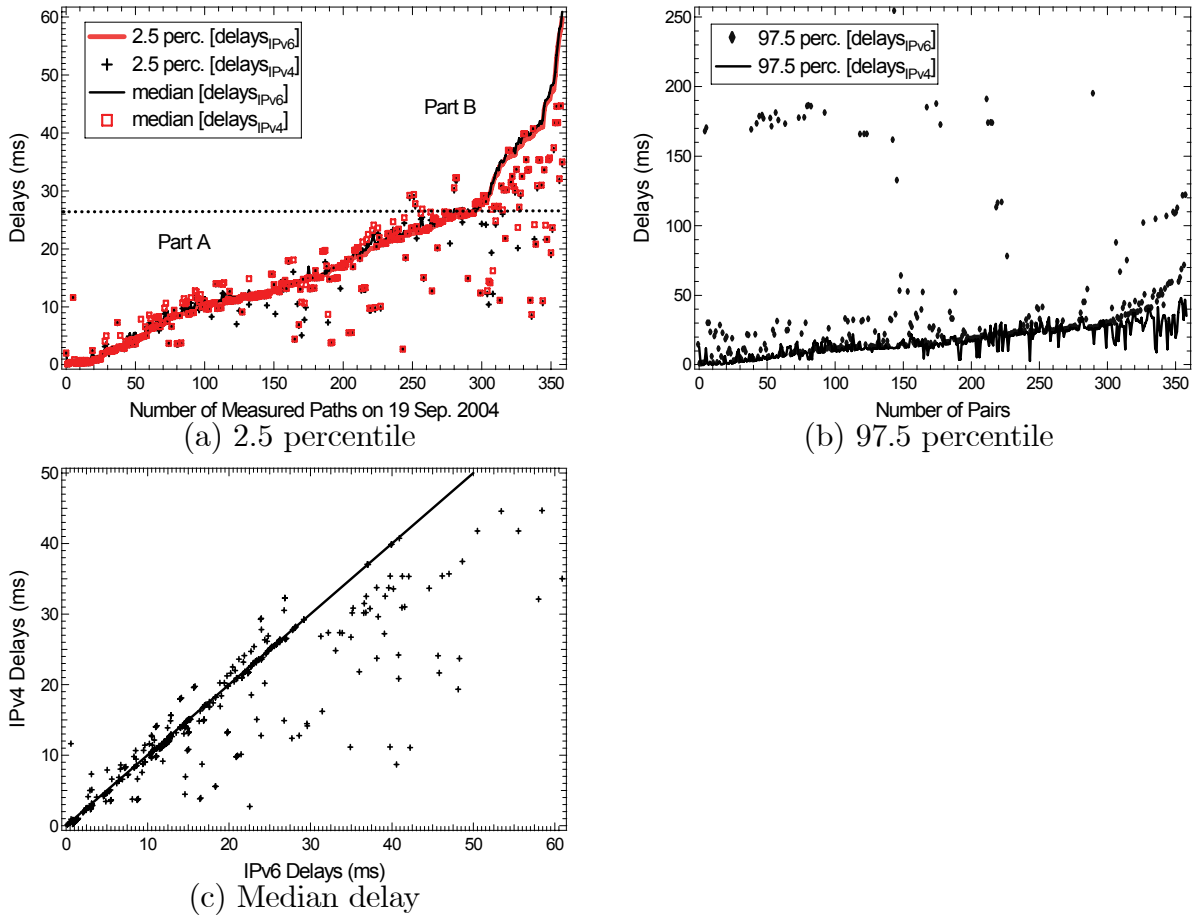


Figure 5.7: Delay trend of a native IPv6 path over 2 years

5.4.3 Delay and Loss Performance of all TTM Paths over a Day

To demonstrate how all IPv6 paths perform on a random day, this section shows the delay and loss performance of the IPv6 paths and their IPv4 counterparts on 19 Sept. 2004. For each source-destination path, we have specifically collected delay and loss performance over that day. We then studied the different behavior of IPv6 native paths and IPv6 tunnel paths by comparing their delay performance with their IPv4 counterparts on a path-by-path basis.

In that day, there were 359 IPv6 native paths, and 181 IPv6 tunnel paths. The native paths were all within a continent (Europe, USA), while about 28% of the tunnel paths were inter-continental (Europe-JP). We observed the following performance difference between native IPv6 paths and IPv6 paths with tunnels.



(a) 2.5 percentile

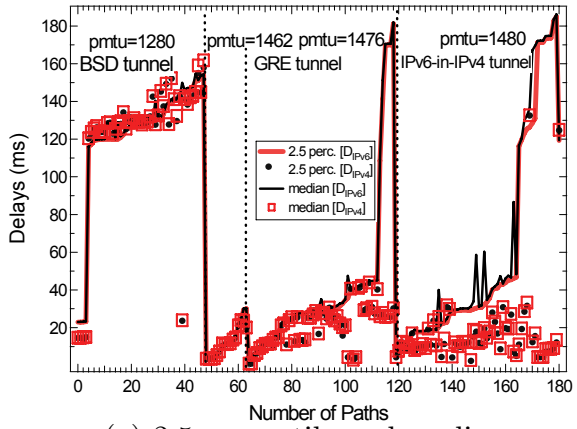
(b) 97.5 percentile

(c) Median delay

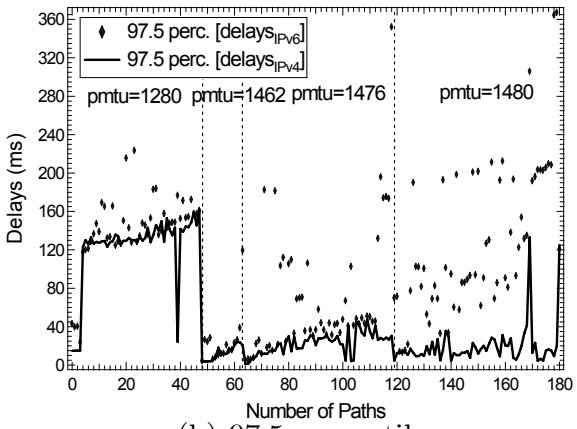
Figure 5.8: Delay performance of (a) the (sorted) 2.5 percentile and median, (b) 97.5 percentile delays, and (c) scatter plot

Native paths delay performance

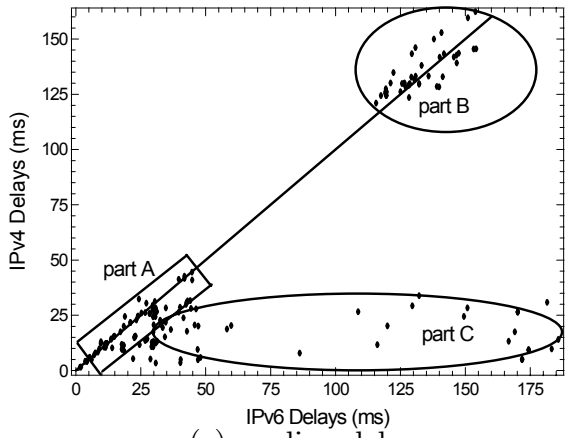
- Figure 5.8.(a) shows the plots of 2.5 percentile and median delay of the IPv6 native paths over that day. For simplicity, we have sorted the paths based on the IPv6 median delay. Figure 5.8.(a) indicates that IPv6 paths with a relative small end-to-end delay (delay $\leq$ 26.5 ms, shown in part A) have comparable 2.5 percentile and median delay to those of their IPv4 counterparts, while those IPv6 paths (in part B) with a relative large end-to-end delay (delay $\geq$ 26.5 ms) suffer a larger delay than their IPv4 counterparts. For 90% of the IPv6 paths, the 2.5 percentile delay is less than 36.1 ms, and the maximum 2.5 percentile delay of the paths is 59.7 ms, while 29.2 ms and 44.6 ms for the IPv4 2.5 percentile delays, respectively. For 90% of IPv6 paths, the median delay is less than 37.1 ms, and



(a) 2.5 percentile and median



(b) 97.5 percentile



(c) median delay

Figure 5.9: (a) The 2.5 percentile and median and (b) 97.5 percentile delays in the IPv6 tunnel pairs and IPv4 counterparts; (c) scatter plot of the median delays in IPv6 tunnel paths and IPv4 counterparts

the maximum median delay of the paths is 60.9 ms, while 30.2 ms and 44.7 ms for the IPv4 counterparts, respectively.

- Figure 5.8.(b) shows that in most cases, IPv6 paths have a larger 97.5 percentile delay. Some IPv6 paths even had a much higher delay. When looking closer at these paths, we found that all these paths contain a site located in Hungary. It was also found that these site-specific effects can change completely later. For 90% of the IPv6 paths, the 97.5 percentile delay is less than 94.3 ms, and the maximum 97.5 percentile delay of the paths is 115.6 ms, and much less (31.1 ms and 48 ms, respectively) for the IPv4 counterparts.

- Figure 5.8.(c) shows the scatter plots of the native IPv6 median delays versus the IPv4 delays, where IPv6 delay is on the X-axis and IPv4 delay on the Y-axis. Each data point corresponds to a path. We also plot the diagonal line. For points below this line, IPv4 outperforms IPv6. The key observation is that most IPv6 paths perform similar to their IPv4 counterparts, some perform worse, but some better. This result agrees with that of Figure 5.8.(a).

Tunnel paths delay performance

- Figure 5.9.(a) shows 2.5 percentile and median delays of the IPv6 tunnel paths and the corresponding IPv4 paths. For clarity, we have classified the paths based on the IPv6 tunnel types (which were deduced from the path MTUs). The results suggest that IPv6-in-IPv4 tunnel paths had a much larger delay than their IPv4 counterparts, while packets using other tunnel types show only slightly worse behavior than their IPv4 counterparts: for 90% of the IPv6 tunnel paths, the 2.5 percentile delay is less than 146.8 ms, and the maximum 2.5 percentile delay of the paths is 184.9 ms, while it is 132.3 ms and 157.9 ms for the IPv4 2.5 percentile delays, respectively. For 90% of the IPv6 tunnel paths, the median delay is less than 151.4 ms, and the maximum median delay of the paths is 186.2 ms, and 132.6 ms and 158.2 ms for the IPv4 counterparts respectively.

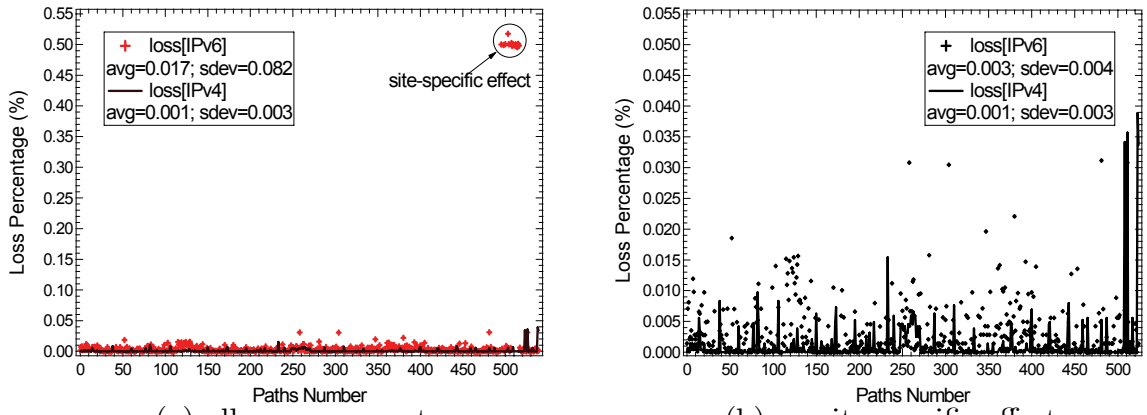
- Figure 5.9.(b) shows 97.5 percentile delay of the IPv6 tunnel paths and the corresponding IPv4 values. The results indicate that all the IPv6 tunnel paths had a larger 97.5 percentile delays than their corresponding IPv4 paths. The worst performance was on paths with IPv6-in-IPv4 tunnels. In general, for 90% of the IPv6 tunnel paths, the 97.5 percentile delay is less than 196.3 ms, and the maximum 97.5 percentile delay of the paths is 367.4 ms, and 135.1 ms and 162.1 ms for the IPv4 counterparts respectively.

- Figure 5.9.(c) shows the scatter plot of the IPv6 tunnel path median delays versus the IPv4 median delays. The data points are broadly classified into three groups

by R, the ratio of the IPv6 over the IPv4 one-way delay: group A for the paths with equal R ($R \leq 1.25$) within the same continent; group B for the paths with equal R ($R \leq 1.25$) between different continents; and group C for the paths with large R ($R > 1.25$). Our results show that for both the IPv6 native paths and tunnel paths, about half of the paths have larger median delays than those of their IPv4 counterparts. For example, for the native paths, about 53% of the paths are of group A, and 47% of group C; while for the IPv6 tunnel paths, 1.6% of paths are of group A, 50.4% of group B, and 48% of group C.

In short, we found that in comparison with their IPv4 counterparts, IPv6 paths with tunnels perform much worse than the IPv6 native paths. The worst performance is found with IPv6 paths that contain IPv6-in-IPv4 tunnels.

IPv6 paths loss performance



(a) all measurement (b) no site-specific effect

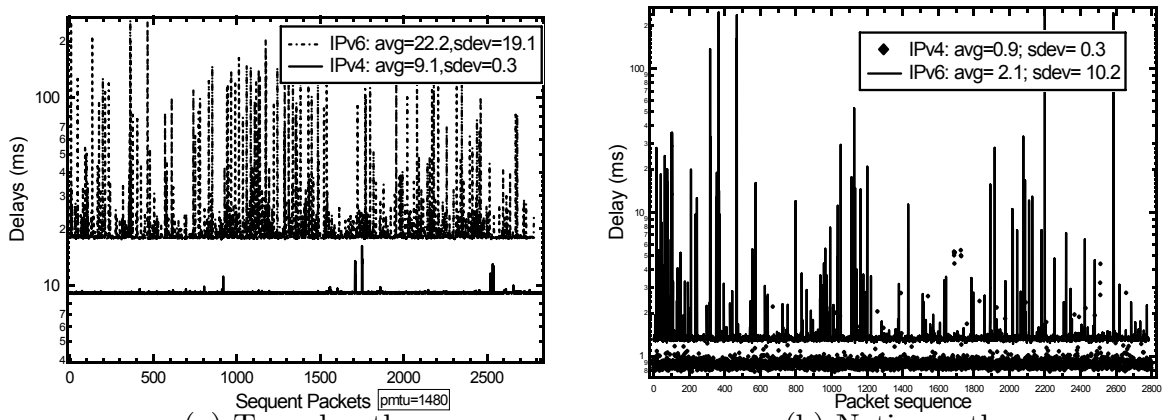
Figure 5.10: Loss percentages of all the paths over a day

Figure 5.10 shows the measured loss results of IPv6 traffic and IPv4 traffic over one day (19 Sep. 2004). The results indicate that in most cases, IPv6 packets suffered a little higher loss than those IPv4 counterparts. For 90% of the IPv6 native paths, the packet loss is less than 0.009%, and the maximum loss of native paths is 0.51%, while 0.004% and 0.039% for the IPv4 counterparts, respectively. For 90% of the IPv6 tunnel paths, the packet loss is less than 0.35%, and the maximum loss of the paths is 0.50%, while 0.002% and 0.008% for the IPv4 counterparts, respectively. Some IPv6 paths had a high packet loss, and all the paths included a site located in Szeged, Hungary. However, some other paths did not experience such high loss rate. We suspect that the difference in loss may be caused by routing problems of the nodes on that site. Besides,

not all routers on the market choose to put the packets into the software when they cannot handle in hardware. In such cases, the packets are simply dropped [84]. Similar site-specific behavior has also been found by Wang *et al.* [105]. We do not argue that the site-specific behavior reflects general Internet behavior, since it has been found that this site-specific effect completely changed at 13 October 2004.

5.4.4 Delay of Single Source-Destination Path over a Day

This section studies the delay performance of a typical IPv6 tunnel path and a typical IPv6 native path over one day. First, Figure 5.11.(a) shows the delay performance of a typical IPv6 tunnel path and its IPv4 counterpart (from a testbox located in Amsterdam to a testbox located in Dublin) in Sep. 2004, where delay is on the Y-axis and the packet sequence number on the X-axis. The tunnel discovery tool detects that IPv6 packets were transferred in an IPv6-in-IPv4 tunnel (MTU 1480) from the source to the destination. Figure 5.11.(a) illustrates that, with regard to this path, IPv6 packets have a larger delay (the mean value of IPv6 delay is 22.19 ms, while that of IPv4 is 9.12 ms) as well as a much larger delay variation.



(a) Tunnel path (b) Native path

Figure 5.11: Tunnel and native IPv6 path delays performance over a day (19 Sep. 2004)

Second, Figure 5.11.(b) shows the delay performance of a typical IPv6 native path and its IPv4 counterpart (both testboxes are located in Amsterdam) over the same day. The corresponding traceroutes show that IPv4 and IPv6 have the same AS path. Once again, we observed that similar high peaks appear in the IPv6 performance. In general, it is expected that IPv6 and IPv4 will compete for the computing resources. Higher IPv6 average delay might confirm that IPv6 packets do have a lower priority than IPv4 ones, and suffer a long delay in the processing time.

Generally, it may be expected the current IPv6 packets using 6in4 tunnels have higher delays than IPv4 packets because tunnels produce additional overheads in the packet size and processing time on the gateway. However, this does not explain the large fluctuation of the delay in the graph. This may have many causes, including queue length variations and variations in the processing time needed to handle the packets in the routers. While IPv4 routing is mostly done in hardware, IPv6 packets more often suffer from a larger variation in the processing time due software routing. It is also possible that IPv6 is given lower priority than IPv4 by ISPs as IPv4 still is much more important for them. Further, IPv6 and IPv4 packets may take different paths between the source and the destination, including IPv6 paths that do not have tunnels. This might be due to the different peering agreements for IPv4 and IPv6. The traceroute results in Figure 5.12 show both IP level and AS level routing for a native IPv6 path and its corresponding IPv4 path: IPv6 behaves different from IPv4: at hop 3 and hop 4, IPv6 reaches a different AS than IPv4. At hop 5, their traceroutes reached the same AS again. Having analyzed the traceroute for other paths, we found that it is common for IPv6 paths to go through different ASes than IPv4. However, without the global information (such as link weights), it is difficult to tell if IPv6 follows a less optimal routing path compared to their IPv4 counterparts.

5.4.5 Measurement Conclusions and Analysis

In terms of the delays over one day, native IPv6 paths have small 2.5 percentile and median end-to-end delay, and delay comparable to that of their IPv4 counterparts. IPv6 tunnels paths have relatively large 2.5 percentile and median end-to-end delay, and about half of the paths have significantly more delay compared to their IPv4 counterparts. The worst performance came from IPv6-in-IPv4 tunnels. For the 97.5 percentile delay, IPv4 far outperforms IPv6 for both native and tunnel IPv6 paths. Delay can be summarized as follows:

$$\text{IPv4 delay} \leq \text{IPv6 native delay} << \text{IPv6 tunnels delay}$$

In terms of loss measurement over a day, IPv4 performs best, and native IPv6 outperforms IPv6 tunnels, but not greatly. Packet loss can be summarized as:

$$\text{IPv4 loss} \leq \text{IPv6 native loss} \leq \text{IPv6 tunnels loss}$$

Unlike in the case of IPv4 counterparts, the delay performance of both native IPv6 and IPv6 tunnels do not change a lot over time.

We attempt to explain the measurements as follows:

1. The extra header in the IPv6-in-IPv4 tunnel may be negligible. Indeed, in store-and-forward, if we assume $10Mbps$ links (which is low), the extra header amounts

IPv6			
hops	IP address	Host name	AS num(s)
1	2001:610:240:2::1	gw6.e01.ripe.net	3333
2	2001:7f8:1::a501:2702:1	e0-0-0.6b2.AMS7.Alter.net	1200/5417
3	2001:600:8:5::2	intermax-ipv6.customer.alter.net	12702
4	2001:600:4:8e5::2	tu.6r001.cwt.esat.net	12702
5	2001:7c8:2:8::4	fe0-0.6rt501.cwt.esat.net	2110
6	2001:7c8:2:9::3	fe0-0.6rt515.cwt.esat.net	2110
7	2001:7c8:a1:1::c178:c976	tt25.ripe.net	2110
IPv4			
1	193.0.0.238	g0013.nikrtr.ripe.net	3333
2	195.69.144.108	ixp1.nl-ams2.eu.bt.net	1200/5417
3	166.49.163.189	t2a1-ge8-0.nl-ams2.eu.bt.net	5400
4	166.49.153.130	166-49-153-130.eu.bt.net	5400
5	193.95.129.19	vlan3.rt002.cwt.esat-x.com	2110
6	193.95.130.154	vlan53.rt501.cwt.esat.net	2110
7	193.95.130.242	vlan515.rt515.cwt.esat.net	2110
8	193.120.201.118	tt25.ripe.net	2110

Figure 5.12: Traceroute information between a source-destination path in both IPv6 and IPv4 protocols

to $\frac{20\times 8 \text{ bit}}{10Mbps} = \sim 16\mu s$ extra delay in each hop. Thus, even if the number of hops in the tunnel is large, this is still negligible.

2. The lack of routers with IPv6 hardware-optimized implementation. Hardware-based IPv6 tunneling implementations are virtually non-existent. However, [84] shows that software-based routing can perform quite well.

3. Other possible causes for the longer delay for IPv6 are:

   - Fewer optimal paths are used for IPv6, especially when tunnels are used.
   - Different or fewer peering agreements for IPv6 may also contribute.
   - Network management and monitoring of IPv6 networks are not as advanced as for IPv4 networks. ISPs do not invest equally in IPv6 network management as they do on IPv4. Also, the experience with IPv4 is greater (*e.g.* more traffic engineering).
   - IPv6 may have a lower priority in the routers, it may be seen as experimental and not be allowed to degrade the performance of the more important IPv4 traffic.

5.5 Related Work

In this section we present work related to the comparison of the performance of IPv4 and IPv6. However, to the best of our knowledge, hardly any work has attempted to quantify the IPv6 performance in the long term.

Srivastava *et al.* [93] describe the implementation of an IPv6 testbed and the interconnection between three domains using IPv6-in-IPv4 static tunnels. They investigated performance issues (like throughput, packet loss and delay) of aviation applications (such as Controller to Pilot Data Link Communication) using Diffserv on an IPv6 based backbone network. Their results suggest that Diffserv implementation and support in IPv6 has matured enough to provide stable and reliable QoS for the aviation applications.

Adam *et al.* [2] analyzed the issues of the implementation of the IPv6 service, IPv6 performance (in the context of a high-speed network), the advantages of current transition technologies, and the problems encountered. They also provided a performance comparison between three different transition mechanisms: IPv6 in IPv4 tunneling, 6PE tunneling (IPv6 over an IPv4 MPLS network), and dual stack in a local very high-speed broadband network. Their experiments indicated that the current dual-stack approach already achieves good performance.

Cho *et al.* [20] measured both IPv6 and IPv4 round-trip delays from two locations. Their results show that the majority of IPv6 paths have delay characteristics comparable to those of IPv4.

In [110], we compared and analyzed the hopcount and end-to-end delay of IPv6 and IPv4 over a month.

5.6 Chapter Summary

In this chapter, we have presented a detailed measurement study based on the RIPE infrastructure and an analysis of the delay and loss evolution of IPv6 networks.

Currently, IPv6 is moving towards commercial deployment, and knowledge of its performance may be crucial for ISPs to understand how to provide a high quality IPv6 network for future Internet applications. We have shown that the average delay of IPv6 is only slightly worse than or comparable to IPv4, while the delay variance of IPv6 is clearly worse. The loss is roughly the same. When looking at the evolution of IPv6 over time, we found that since the October 2003, the median IPv6 delay has reduced from about 120 ms in Oct. 2003 to about 55 ms in Oct. 2005. Packet loss in both IPv6 and IPv4 has remained small over the two year period.

It is expected that IPv4 and IPv6 will coexist for a while, and that IPv6 tunnels will play a key role in the transitional phase. Software-based routers and tunnels provide a quick way to deploy IPv6. The drawback is that IPv6 tunneling degrades the traffic performance, mainly in terms of larger delay. Clearly, our results suggest that for a better IPv6 quality, only native IPv6 and hardware-based routers should be used everywhere. Still, the performance quality of software-based routers and tunnels is still acceptable for a successful transition phase.

Of course, comparative studies of IPv6 networks in North American or East Asia will help illuminate the IPv6 packet loss and delay specific to their environments.

Chapter 6

Estimation of Voice over IP Quality in the Netherlands

To end-users, it is not the network performance (Chapter 3, 4, and 5) that matters most but the perceived quality of applications running over the network. In general it can be stated that the large scale deployment of applications will only be successful if the perceived quality of these applications is sufficiently high. We thus investigate how the application qualities perceived by the end users (at the application layer) are influenced by the network-layer performance. Voice over IP (VoIP) and Peer-to-Peer (P2P) are two applications that have recently attracted a lot of attention due to their wide deployment. This chapter thus investigates the estimation of the VoIP perceived quality, and Chapter 7 investigates the estimation of P2P distance through real Internet measurement.

6.1 Problem Statement

Voice over IP (VoIP) is becoming an increasingly popular and a cheap alternative to public switched telephone networks (PSTNs). VoIP technology also enables the integration of both data and voice traffic in the same network; allows easy introduction of new multimedia services, and supports more flexibility in terms of codecs. For example, PSTNs are bound to a single codec G.711, while VoIP can use any codec supported by both user terminals. However, due to the connectionless, packet-switched character of IP networks, packets may experience different delay, arrive at the destination out-of-order or even get lost. All of these (*i.e.* different delay, packet reordering, and packet loss) affect the perceived quality of voice calls.

The Internet is made up of a large number of separate networks interconnected at exchange hubs. If a packet is sent from one network to another, it has to pass through

R-value rang	100>R>90	90>R>80	80>R>70	70>R>60	60>R>0
MOS	4.50-4.34	4.34-4.03	4.03-3.60	3.60-3.10	3.10-1.00
Speech quality	best	high	medium	low	very poor

Table 6.1: Speech transmission quality classes and corresponding R-value ranges

one of these hubs. There are four high-speed hubs in The Netherlands[1], the biggest being the Amsterdam Internet Exchange (AMS-IX). Since interactive services such as VoIP are not only increasingly important but also impose stringent requirements to the network, assessing the performance of VoIP is an important issue. Many Dutch Internet operators offer services to small and medium enterprises. They provide email, Internet access with firewall, Windows networking and backup services, as well as national VoIP.

This chapter describes an assessment of VoIP quality in the Netherlands, and the network performance measured for VoIP packets sent between 12 Internet testboxes (servers) of a Dutch ISP. Our observations are based on packet level traces collected throughout the network. The main aim is to understand to what extent today's Internet (in the Netherlands) meets the quality requirements for voice calls from the perspective of users.

Several researchers have worked on the measurement and assessment of VoIP quality over Internet. The one whose work is closest to ours Marsh *et al.* [65], who measured VoIP quality on an hourly basis by tracing a pre-recorded PCM coded call between nine sites in 2002, and comparing the results with those obtained from a similar study in 1998. Their results showed that the best-effort Internet is sufficient for VoIP. Our work differs from [65] in terms of the experimental setup since we analyzed real network traces using far more different encoding schemes (up to 6). In addition, we considered the impact of the playout buffer.

6.2 Prediction of the Voice Quality with E-Model

The E-Model [49] was used to estimate the subjective quality of voice calls. According to ITU-T Recommendation G.107, every rating R-value calculated from the E-Model corresponds to a Mean Opinion Score (MOS) value (Table 6.1) to predict subjective user reactions. An R-value above 70 corresponds to PSTN quality.

The mapping function from an R-value to a MOS value has the following form [23]:

$$MOS = 1 + 0.035R + 7 \times 10^{-6} R(R-60)(100-R) \tag{6.1}$$

[1]The Amsterdam Internet Exchange (AMS-IX) in Amsterdam, the Netherlands Internet Exchange (NL-IX), also in Amsterdam, the Groningen Internet Exchange (GN-IX) in Groningen and the Dutch-German Internet Exchange (ND-IX) in Enschede.

Parameters	G.711	G.729(10ms)	G.729(20ms)	G.723.1	iLBC
bitrate(kb/s)/framesize(ms)	64/20	8/10	8/20	6.3/30	15.2/20
a	0	10	10	15	10
b	30	25.21	25.21	36.59	19.8
c	15	15	20.2	6	29.7

Table 6.2: Parameters for different codecs (except for GSM)

where the output of the E-Model is the rating factor R:

$$R = (R_0 - I_s) - I_d - I_e + A \tag{6.2}$$

where R_0 is the effect of background and circuit noise, while I_s captures the effect of quantization. Both R_0 and I_s describe the transmitted voice signal itself and do not depend on the transport network. I_d is the impairment caused by one-way delay of the path, and I_e is the impairment caused by losses. A is the expectation factor. Based on recommended values in [49], the rating R can be defined by

$$R = 94.2 - I_d - I_e \tag{6.3}$$

where I_d has the following form:

$$I_d = 0.024d + 0.11(d - 177.3)H(d - 177.3) \tag{6.4}$$

where d is the one-way delay in milliseconds, and $H(x)$ is the Heavyside or step function where $H(x) = 0$ if $x < 0$ and 1 otherwise.

Unlike I_d, which only depends on the transport network and not on the codecs, I_e is codec dependent. The following form is presented in [23]:

$$I_e = a + b\ln(1 + cP/100) \tag{6.5}$$

where P is the packet loss rate in percentages, while a, b and c are fitting parameters for various codecs [48].

The specific values of a, b and c for different codecs (except for GSM) are shown in Table 6.2. For G.711, it is assumed that Packet Loss Concealment has been implemented. The codec iLBC (internet Low Bitrate Codec) [39] is a free speech codec suitable for robust voice communication over IP networks. The parameter values for G.729 and G.723.1 are derived in [31][30], while the values for G.711 are derived in [23]. To calculate the a, b and c for iLBC, we extracted the iLBC MOS versus P from GlobalIPsound [39], then converted this relationship to I_e versus P via (6.1) and (6.2). The fitting model for the iLBC codec is shown in Figure 6.1. Note that G.729 and iLBC have the same I_e values if there is no packet loss. Thus we took the same a value for iLBC as that of G.729.

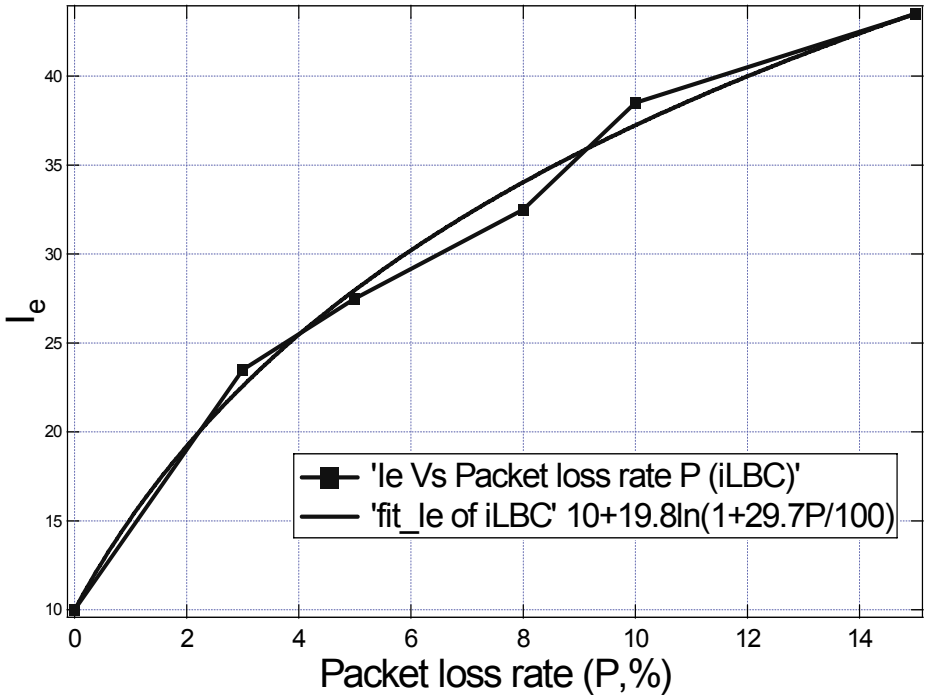


Figure 6.1: I_e vs. packet loss rate P for iLBC

For GSM (13 kbit/sec and 22.5 ms), ITU-T G.107 [49] and G.113 appendix I [50] were used. The corresponding formula[2] for I_e is:

$$I_e = 5 + 90\frac{P}{P+10} \tag{6.6}$$

6.3 Experiment Results

The locations of the twelve testboxes have been chosen uniformly over the area of the Netherlands. They are mainly 2 or 3 hops away from the high speed backbone network, and their locations are shown in the map of Figure 6.2. The sites were connected in a full mesh. The terminal clocks were synchronized using NTP software (with an accuracy of about ± 3 ms) every half an hour. Different encoding schemes were used. The packet sizes were calculated for different codecs. Following ITU-T P.59 recommendation [98], a sequence of alternating voice signals and silence periods (without hangover time) was

[2]However, only values for GSM 6.60 Enhanced Full Rate (EFR) are given, which has a slightly lower bit rate than the simulated packet streams (12.2 kbit/sec instead of 13 kbit/sec) that were based on GSM 6.10. There, therefore, is a small inconsistency here.

Figure 6.2: Locations of the test-boxes (black boxes) in the Netherlands

used as an input signal. No voice packets were generated during silence periods.

The 12 testboxes participated in two experiments. First, during a 2 week period from Feb. 2, 2005 to Feb. 15, 2005, between each sender-destination pair of measurement boxes, packets generated in parallel with different codecs G.729, G.723.1 and GSM, were continuously transmitted from 7 AM to 9 PM (Local Time). Second, we repeated the same experiment with G.729, G.711 and iLBC packets over 10 days from June 3, 2005 to June 12, 2005. The difference between the two experiments may indicate how Internet packet dynamics change over time.

During the experiments, about 10 gigabytes experimental data (such as sending time, arrival time, and sequence numbers) were collected at a central point. A packet was classified as a reordered or out-of-order packet if it had a sequence number smaller than its predecessors. We examined each arriving packet by checking its arrival sequence order to calculate the total number of reordered packets.

We also executed traceroutes every 6 minutes during each test to determine the route taken during the tests. The resulting lists of intermediate routers of the paths were checked with the RIPE database [88]. The traceroutes provide some insight into the structure of the Internet in The Netherlands and they are useful to verify the

changes in the delay during the measurements. Our results indicate that almost all traffic (99.2%) is routed through the AMS-IX. Of the remaining 0.78% the routing is unclear. Only a few routes are very inefficient. Traffic on these routes is either routed through a router in London or through a router in Frankfurt (via the AMS-IX).

6.3.1 Network Delay Performance

It is well-known that VoIP will not perform well if delays between the communicating parties exceed a certain QoS delay threshold (*i.e.* 150 ms). In this section, we will discuss the delay measured between the 12 testboxes. Figure 6.3 summarizes the complementary cumulative distribution function (CCDF) of the 97.5 percentile delays with different codecs in our experiments. Each data point corresponds to a pair of peers. Our results indicate that packets experience low delays. About 95% of all experimental pairs have a 97.5 percentile delays less than 74 ms. For the completeness, here we also present the results of the median, average, and 99 percentile delays with different codecs in our experiments. About 95% of all experimental pairs have median delays less than 40 ms, and the same holds for the average delays, and 95% of all experimental pairs have a 99 percentile delays less than 114 ms. Moreover, we also observe that those delay distributions do not vary significantly from codec to codec.

The end-to-end delay for voice over PSTN is less than 100 ms (reported by U. Varshney *et al.* [104]). Our results of the voice delays in the Netherlands are lower than those reported for both voice over IP and voice over PSTN in [104].

We observe that few paths suffer from large delay, and these are mainly caused by heavy load (with large packet loss) at the links connecting these pairs in the rush hours, and system updates in the testboxes.

Figure 6.3 shows that the CCDFs of the delays D exhibit heavy tails: most individual pairs have a relatively small delay, but large outliers are not uncommon. This suggests that networks in the Netherlands in 2005 can achieve high performance. The heavy tail is fitted by a power law defined as $\Pr[D > x] \simeq cx^{-b}$, where the number b is the power law exponent (*i.e.* the slope in a log-log plot). Figure 6.3 shows the exponents $2.45 \leq b \leq 4.68$ in the distributions of the medians delay, while $2.53 \leq b \leq 4.17$ in the average delays, $1.59 \leq b \leq 2.24$ in the 97.5 percentile delays, $b \approx 1.41$ in the 99 percentile delays.

6.3.2 Network Packet Loss Percentage

The packet loss percentage P is the percentage of unreceived packets in the data network. Unlike applications like email or ftp, which can simply request retransmission when data is lost, VoIP discards those voice samples that are lost or arrive too late. Packet loss results in a degradation of the conversational voice quality. According to industry standards, the maximum packet loss tolerable is about 3%.

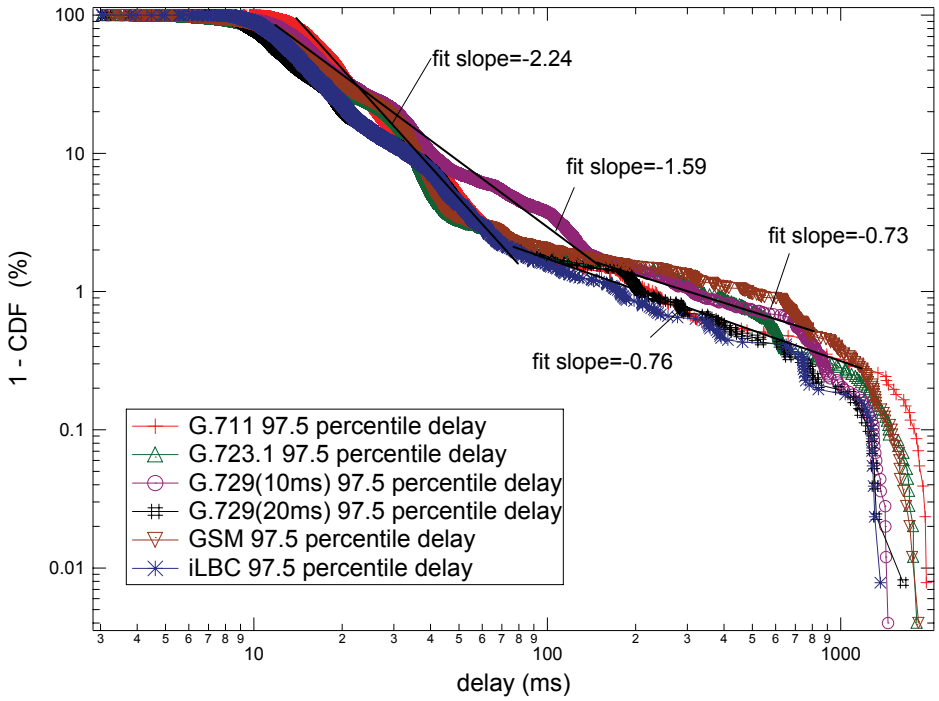


Figure 6.3: The CCDFs of the 97.5 percentile delays with different codecs in our experiments, and their corresponding power law fits.

Figure 6.4 plots the CCDF of the percentage of lost packets. The CCDFs of the packet loss exhibit very heavy tails: This suggests that most (above 70%) individual pairs for different codecs have virtually no packets loss, while about 99.5% of all the pairs for different codecs have the percentage of packet loss less than 1%. We also observed that few paths suffered from large packet loss ($> 5\%$), and this is mainly caused by the system updates of the testboxes. The experimental results suggest that in our experiments, the packet loss is low enough to satisfy the industry standards.

6.3.3 Reordering Packets

Reordering of packets may impact the performance of applications on the Internet. In a TCP connection, the reordering of three or more packets within a flow may cause fast retransmission and fast recovery multiple times resulting in a reduced TCP window size and consequently in less throughput for the application. For delay-sensitive services which use UDP as transport protocol (such as VoIP or video conference), the ability to restore the order of packets at the destination has finite limits. The deployment of a real-time service necessitates certain reordering constraints to be met. For example, in

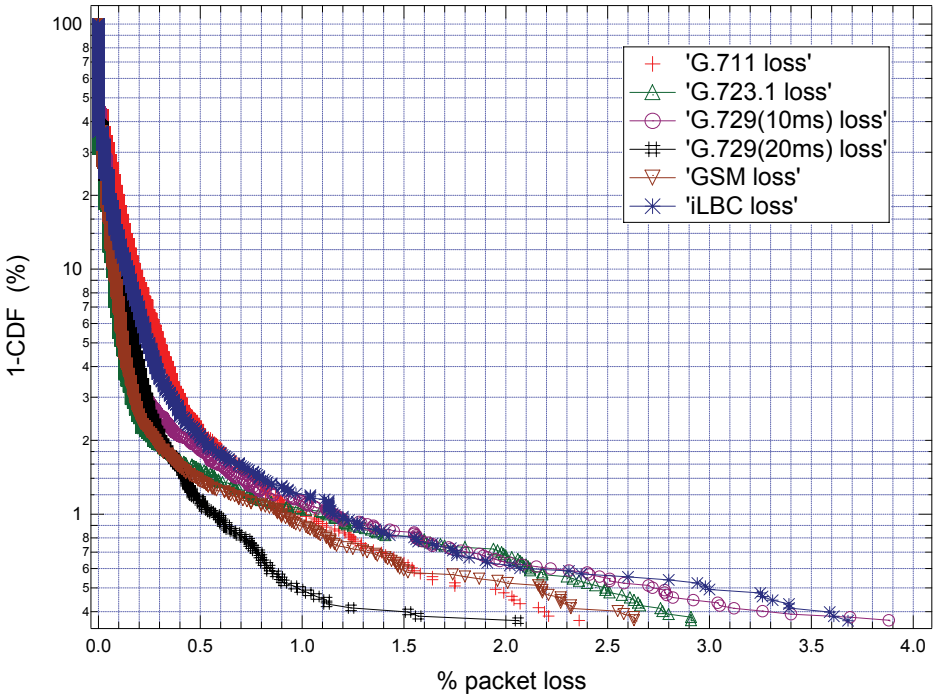


Figure 6.4: The CCDF of the packet loss percentage.

codecs	G.711	G.723.1	G.729(10ms)	G.729(20ms)	GSM	iLBC
Total Nr. of reordering	3	2	2	4	7	1

Table 6.3: Total number of reordered packets received (source)

case of VoIP, to maintain the high quality of voice, packets need to be received in order, and also within 150 ms. To verify whether these QoS requirements can be satisfied, knowledge about reordering in the Internet is desirable. To measure the number of reordered packets, for each source-destination pair with different codecs, we examined each arriving packet by checking its arrival sequence order, and calculated the total number of reordered packets for different codecs by summarizing the reordered packets for different codecs measurement.

Table 6.3 shows the total number of reordered packets observed for different codecs during the period of our experiments. Of all the packets sent successfully in our experiments, only very few (<0.0001%) are reordered. This result suggests that reordering is negligible when the network load is relatively light (*e.g.* the bit rate of the packets is below 64 kb/s).

6.3.4 Estimation of the Voice Quality

In order to apply the E-model to assess the perceived voice quality, we need to estimate end-to-end delay and packet loss. Both end-to-end delay and packet loss consist of a part induced by the network and a part originating in the VoIP terminals. Network performance has been discussed in the previous sections. The sending terminal contributes to the end-to-end delay through packetisation and coding delay. Typical values per codec can be found in [50]. The receiving terminal also adds delay through the operation of the playout buffer at the receiver side, which compensates the effects of delay jitter by holding the first packet in a voice call for some time T before it is decoded. The dejittering delay T adds to the end-to-end delay. In this thesis we assume that T is fixed at $40ms$. Packets that arrive too late in the playout buffer to be decoded are considered lost. Hence, the playout buffer also contributes to the end-to-end packet loss. In our case the packet loss ratio induced by the playout buffer equals the ratio of packets that experience a network delay exceeding the minimum delay plus $40ms$. We now apply the E-model. Figure 6.5 shows the voice call ratings and MOS values for different codecs in our two experiments. From Figure 6.5 we can see that G.711 gives the highest call rating, followed by GSM, while the G723.1 gives the lowest call rating. The results for iLBC are almost as good as G.729. In general, the quality of calls in different codecs is very high: with 99% of all calls experiencing a quality above 74 (3.7 in MOS). These results confirm that high VoIP quality can be achieved in the Netherlands. However, few paths achieved low MOS value ($MOS < 3.7$) due to high delay and loss.

To determine the time variation in the quality of the calls, we calculated the average delay, average packet loss rate, R and MOS values for every 1.5 hours of daily experiment (thus 6 sample points per day).

Figure 6.6 shows these values versus time. Figure 6.6(a) shows that all the packets in different codecs had a very low end-to-end delay, mainly in a range of 9-25 ms (below the noticeable 100-150 ms). There are higher delays in the rush hours compared with morning (7 AM-8 AM) and night (7:30 PM-9 PM). The corresponding traceroutes indicate that most source-destination pairs followed fixed paths, indicating that the delay variation may be caused by queueing. Figure 6.6(b) shows average packet loss versus time. The experimental results indicate that almost all the pairs experience consistently very small (or even no) loss ratios during our two experimental periods. Figure 6.6(c) and Figure 6.6(d) indicate that the current Internet in the Netherlands can continuously achieve satisfactory results ($MOS \geq 3.7$). Our measurements suggest that the paths with low delay and loss can achieve an excellent MOS ($4 \leq MOS < 4.4$) at all times except for the rare cases when outages occur (*i.e.* system updates in the testboxes). We repeated the experiments by calculating the average delay, average packet loss rate, R and MOS values for a smaller time scale (1 minute voice of daily experiment) and observed similar results, which are not shown here.

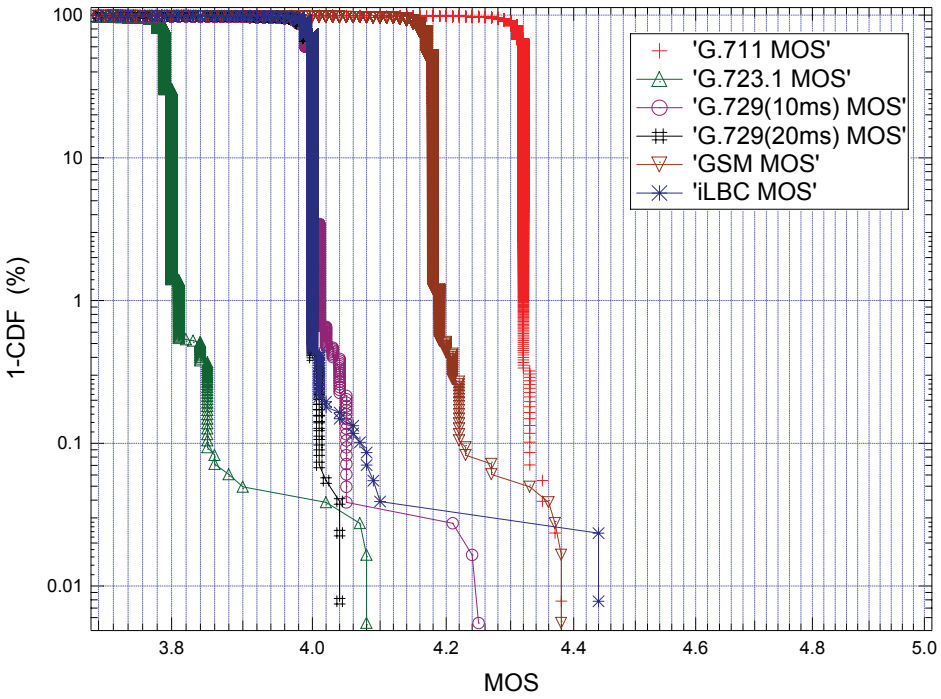


Figure 6.5: The CCDF of call quality for different codecs

6.4 Chapter Summary

In this chapter, we have estimated the quality of VoIP as experienced by users by tracing UDP packets sent between 12 testboxes in the Netherlands. Our results lead to several observations with respect to network and VoIP in the Netherlands:

- Packet reordering hardly ever occurs.
- Paths can continuously achieve low delay and low packet loss.
- Networks can continuously support satisfactory VoIP quality.

An important aspect of future work will be the study of the impact of perceived VoIP quality in a larger network environment. The RIPE TTM [87] infrastructure, which consists of about 100 testboxes located in different countries, can be used for this purpose. We also want to study the relation between the quality as experienced by users on different time scales, *e.g.* on an hourly basis and based upon measurements averaged over 1 minute. The impact of different playout buffer schemes (like fixed playout and adaptive scheme) will also be a subject of future work.

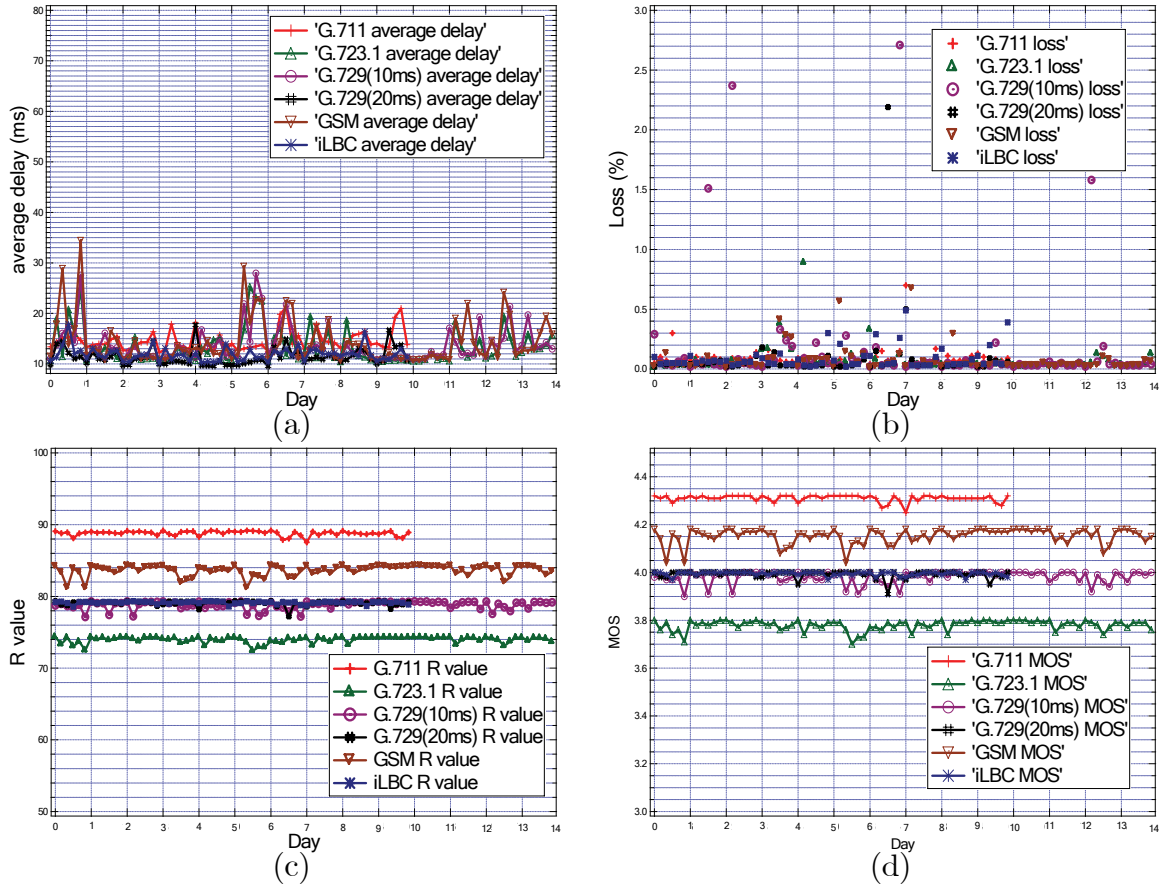


Figure 6.6: Call quality statistics for different codecs for every 1.5 hours of each experimental day.

Chapter 7

P2P Distance Estimation

7.1 Introduction

An efficient mechanism to estimate distance in the Internet may help many large-scale distributed network applications such as nearby server selection and peer-to-peer computing. One example of this is when a client has to select the nearest from a set of equivalent servers. Similarly, optimization of overlay networks like peer-to-peer networks often requires nodes to connect to peers in their neighborhood. Usually "separation" or "neighborhood" is defined in terms of the network delay experienced in a packet exchange, but we will use the general term "network distance" to include other quantities as well, in particular, besides delay, we will study the hopcount.

Actively measuring network distances between many pairs of nodes imposes a large load of probe packets on the network. Delay measurement requires at least a few packet exchanges to filter out noise. A hopcount measurement using for instance traceroute is even more expensive. Mapping-techniques based on landmarks or beacons attempt to solve this problem [72][42]. In these techniques each of the N nodes measures its distance to a small set of M well-known landmark servers. By conceiving the results as the components of a vector, each node is embedded in a M-dimensional hyperspace. Now it is assumed that the distance between two network nodes can be estimated by computing the (*e.g.* Euclidean) distance between their respective coordinate vectors in the hyperspace. This assumption implies that the entries of a large $N \times N$ distance matrix can be estimated when knowing only M rows or columns. It is important to stress that the method is entirely heuristic; the only a priori basis is the likelihood that nearby nodes in the network have "nearby" coordinate vectors. This assumption is based on the fact that it seems possible in practice.

This section consists of two experiments: in the first, we evaluated fixed landmark-based distance estimation using various hyperspace distance functions. The motivation behind this experiment is to study which hyperspace distance function correlates best

to the actual distances between hosts. In the second experiment, we investigated the aging of landmark-based coordinates. Since hosts do not continuously update their coordinates, it is interesting to investigate the quality of prediction based on coordinates which have been determined earlier, and to determine how often peers should update their coordinates. This knowledge may help manage network distance estimation.

Our empirical evaluation is based on data from the RIPE NCC TTM project. The network delay directly influences the user experience in many applications. There can be an indirect influence as well because the TCP throughput is roughly inversely proportional to the round-trip time. The hopcount, on the other hand, is not usually a network distance of direct relevance to the user, but is an important quantity from a network-centric point of view. When nodes connect to peers separated by a small number of IP hops, the amount of traffic traversing network boundaries is potentially limited. This argument of "segregating traffic by topology" has also been made in [42], but a detailed account of the efficiency is still not available.

7.2 Problem Description and Definitions

Let $d(A, B)$ be the network distance between node A and B. We assume that $d(A, B) \geq 0$. Note that our "network distance" is not a mathematical distance function. In particular, the triangle inequalities may not hold.

The M landmark servers are denoted by $L_1, ..., L_M$. To each node X_i we assign a coordinate vector

$$\boldsymbol{x}_i \equiv (\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{iM}) \tag{7.1}$$

where $\boldsymbol{x}_{ik} = d(X_i, L_k)$ is the measured distance between the node i and the landmark k. The hyperspace distance between the coordinate vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is written as $D(\boldsymbol{x}_i, \boldsymbol{x}_j)$, to distinguish it from the network distance d. The central question is: To which extent is $D(\boldsymbol{x}_i, \boldsymbol{x}_j)$ correlated with $d(X_i, X_j)$? Here we answer this question for the following functionals:

$$D_q(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{k=1}^{M} (\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk})^q \right)^{1/q}, q > 0 \tag{7.2}$$

$$D_+(\boldsymbol{x}_i, \boldsymbol{x}_j) = \min_{k=1,...,M} (\boldsymbol{x}_{ik} + \boldsymbol{x}_{jk}) \tag{7.3}$$

$$D_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = (D_+ + D_\infty)/2 \tag{7.4}$$

$$D_G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{D_+ D_\infty} \tag{7.5}$$

$D_q(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{k=1}^{M} (\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk})^q \right)^{1/q}, q > 0$ The Holder q-norm D_q generalizes the Euclidean distance D_2 which was used in [58]. For $q = \infty$, D_q reduces to $D_\infty(\boldsymbol{x}_i, \boldsymbol{x}_j) =$

$\max_{k=1,\ldots,M} |\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk}|$. The functions D_∞ and D_+ play a special role in the sense that they are a lower and upper bound for the network distance d, assuming that d satisfies the triangle inequalities. Based on this observation, the arithmetic average D_A of D_∞ and D_+ has been proposed as a hyperspace distance [44][42]. We also include the geometric mean D_G. If d satisfies the triangle inequalities we have

$$D_\infty \leq D_G \leq D_A \leq D_+ \tag{7.6}$$

When using the hyperspace distance $D(t)$ with the data obtained at time t to estimate the network distance $d(t + \Delta t)$ at time $t + \Delta t$ (Δt is called a time lag), the linear correlation coefficients of the predicted hyperspace distance $D(t)$ with the network distance $d(t + \Delta t)$ are defined by

$$\rho(t, \Delta t) = r(D(t), d(t + \Delta t)) \tag{7.7}$$

7.3 Experiment of Landmark-based Distance Estimation

Our data are provided by RIPE NCC TTM project (see Section 2.3). In the first experiment, to evaluate landmark-based distance estimation using various hyperspace distance functions, we analyzed the data collected by RIPE NCC TTM on May 15, 2003, at which time there were 58 active boxes, 48 located in Europe, 7 in the US, and 1 each in Japan, Australia and New Zealand. The map in Figure 2.3 shows the geographical distribution of the test-boxes.

For each sender-destination pair we computed the minimum end-to-end delay over 24 hours (that is approximately 2160 probe-packets), in order to determine the congestion-free delay. The RIPE TTM differs from other infrastructures like PingER and AMP in that it measures one-way delays rather than RTTs. In this thesis, however, we ignore asymmetry, and consider only symmetric network distance, defined as the sum of the network distances in both directions. For delay, the result can be considered as a round-trip time. For hopcount, the result can also be considered as a round-trip distance, and we considered only the most dominant path during 24 hours, omitting pairs for which the delay or hopcount in one of the directions is missing, leaving in total 1503 pairs and 1024 within Europe. These measurements provide us with the network distance matrix $d(A, B)$.

For both experiments, 10 of all the test-boxes were assigned the status of landmark. They were chosen randomly, except that 6 are located in the EU, 1 in the Asia and 3 in the US. The remaining test-boxes are referred to as "peers". We repeated the experiment with $M = 5$ respectively $M = 15$ landmarks.

7.3.1 Delay

Figure 7.1 plots the data for the delay, using 10 landmarks using six panels, corresponding to six different hyperspace norms: D_1, D_2, D_∞, D_+, D_A, and D_G. Each data point corresponds to a pair of peers. The horizontal axis represents their actual network distance $d(A, B)$, the vertical axis their distance predicted by the landmark scheme, that is their hyperspace distance $D(\boldsymbol{x}_A, \boldsymbol{x}_B)$. The lines shown are the least-squares fitted line, and the diagonal $D = d$. The inset also shows the linear correlation coefficient,

$$r(D, d) = \frac{cov(D, d)}{\sqrt{Var(D) \cdot Var(d)}} \tag{7.8}$$

We repeated the experiment using a smaller (5) and a larger (15) number of landmarks. We also repeated them on the subset of test-boxes (landmarks and peers) all in Europe, again with 5, 10, and 15 landmarks. The results are summarized in Table 7.1. Each row corresponds to a different hyperspace distance function; each column corresponds to a different number of landmarks, selected from the complete set of test-boxes ("All") or from those in Europe only ("Eur"). The first number in each cell is the linear correlation coefficient $r(D, d)$. It makes sense to also study Spearman's rank correlation coefficient

$$r'(D, d) \equiv r(rank(D), rank(d)) \tag{7.9}$$

quantifying the ability to predict which nodes are closest irrespective of the actual values of the distance. The rank correlation is the second number in each table cell.

	5 (All)	**10 (All)**	**15 (All)**	**5 (Eur)**	**10 (Eur)**	**15 (Eur)**
D_1	0.86, 0.91	0.91, 0.88	0.89, 0.91	0.76, 0.75	0.89, 0.84	0.64, 0.80
D_2	0.90, 0.91	0.93, 0.86	0.96, 0.90	0.78, 0.74	0.90, 0.80	0.83, 0.74
D_∞	0.93, 0.90	0.91, 0.77	0.88, 0.84	0.65, 0.70	0.70, 0.75	0.91, 0.70
D_+	0.96, 0.90	0.98, 0.93	0.99, 0.93	0.96, 0.80	0.96, 0.87	0.93, 0.84
D_A	0.95, 0.93	0.97, 0.86	0.98, 0.90	0.79, 0.77	0.81, 0.82	0.91, 0.76
D_G	0.94, 0.93	0.97, 0.90	0.98, 0.91	0.81, 0.81	0.77, 0.82	0.94, 0.83

Table 7.1: Correlation coefficients for the delay data.

The D_+ hyperspace distance gives the highest correlations, in particular higher than for the Euclidean distance D_2. The results for D_A and D_G are almost as good as D_+.

7.3.2 Hopcount

We now turn to the results for the hopcount. Figure 7.2 shows the results for 10 landmarks.

The complete results including those for $M = 5$ and $M = 15$ landmarks are summarized in Table 7.2 with the same notation as Table 7.1.

	5 (All)	**10 (All)**	**15 (All)**	**5 (Eur)**	**10 (Eur)**	**15 (Eur)**
D_1	0.23, 0.19	0.45, 0.46	0.51, 0.53	0.39, 0.40	0.45, 0.46	0.57, 0.60
D_2	0.29, 0.28	0.48, 0.47	0.54, 0.54	0.40, 0.37	0.46, 0.46	0.57, 0.60
D_∞	0.35, 0.37	0.41, 0.40	0.47, 0.45	0.40, 0.35	0.38, 0.35	0.46, 0.44
D_+	0.67, 0.65	0.71, 0.67	0.79, 0.78	0.64, 0.58	0.65, 0.60	0.70, 0.62
D_A	0.67, 0.71	0.78, 0.74	0.81, 0.80	0.66, 0.61	0.70, 0.65	0.75, 0.70
D_G	0.54, 0.58	0.71, 0.67	0.76, 0.74	0.55, 0.50	0.62, 0.58	0.68, 0.65

Table 7.2: Correlation coefficients for the hopcount data.

The plots and the table both show that for the hopcount, compared to the delay, the correlation between estimated and actual distance is smaller. The functional D_A shows the largest correlation. Another interesting observation about the hopcount is that the majority of data-points in the scatterplot for D_∞ (D_+) are located below (above) the diagonal. In practice, this means that these lower and upper bounds derived by assuming the triangle inequalities are more often not satisfied. Note that a violation of only one of the inequalities for one of the triangles $X_iX_jL_k$ is sufficient for a violation of either the lower bound (*i.e.* $d < D_\infty$) or the upper bound ($d > D_+$).

7.3.3 On Network Distance Triangulation

We have seen that though the delay and hopcount do not always satisfy the triangle inequalities, a prediction scheme based on the triangle inequalities can be relatively successful. These network distances are apparently sufficiently correlated between triplets of nodes. In this section we try to shed some light on the possible origin of such correlations. The most obvious origin is the fact that a communication network is embedded in the real, Euclidean world. Even when network distances are only weakly correlated to geographical distance, the network distance matrix will bear some trace of the geographical triangle inequalities.

Geographical embedding is, however, not a necessary condition for correlations in the network distance matrix. Consider an arbitrary network topology where paths are sought to minimize the sum of given link weights. Notice that in such a scenario the path weights obey the triangle inequalities by construction, whereas the hopcount and delay do not. This is mainly due to the policy routing. We conducted simulations to determine to what extent both types of network distances can be predicted using landmarks.

For sake of simplicity we simulated random graphs with i.i.d. link weights w. We applied landmark estimation to both the path weight and the hopcount, for the six

hyperspace distances defined before. In particular we generated random graphs of 2000 nodes and link probability $p = 0.01$. Each link weight w is drawn from a polynomial distribution,

$$\Pr[w \leq x] = x^\alpha \cdot 1_{x \in [0,1]}$$

where α is a positive real number. For large values of α the link weights are nearly constant (and equal to unity), so the path weight becomes identical to the hopcount. For $\alpha = 1$ the link weight distribution is uniform. For small values of α the fluctuations of the link weights are much larger than their average. As representative values for these regimes, we consider $\alpha = 0.1$, 1, and 10. For each value of α, 10^4 random graphs were generated and, in each of them, 10 random nodes were selected as landmarks, and 10 others were randomly chosen as "peers". Table 7.3 summarizes the results. The column names "hops" and "weights" correspond to the choice of the network distance being estimated.

	$\alpha = 0.1$		$\alpha = 1$		$\alpha = 10$	
	hops	**weights**	**hops**	**weights**	**hops**	**weights**
D_1	0.60, 0.54	0.97, 0.98	0.13, 0.13	0.92, 0.86	-0.10, -0.10	0.39, 0.33
D_2	0.60, 0.58	0.97, 0.99	0.14, 0.14	0.93, 0.87	-0.10, -0.10	0.47, 0.39
D_∞	0.60, 0.57	0.99, 0.99	0.16, 0.16	0.92, 0.87	0.12, 0.10	0.50, 0.41
D_+	0.60, 0.60	0.97, 0.99	0.20, 0.19	0.98, 0.94	0.32, 0.31	0.65, 0.43
D_A	0.70, 0.71	0.99, 0.99	0.25, 0.24	0.98, 0.95	0.35, 0.33	0.70, 0.51
D_G	0.70, 0.71	0.99, 1.00	0.25, 0.24	0.97, 0.94	0.27, 0.32	0.70, 0.49

Table 7.3: Correlation coefficients for the random graph simulations.

The following observations were made from these data:

- The correlations for "hops" are always smaller than for "weights". This was more or less expected, in view of the fact that the (minimized) path weights satisfy the triangle inequalities, while their hopcounts do not. For larger values of α the difference between hops and weights will disappear, and both will satisfy the triangle inequalities. (The results shown for "hops" and $\alpha = 10$ are somewhat anomalous for the network size studied because the hopcount is only a small integer.)

- For $\alpha = 0.1$ and "weights" the correlations are nearly perfect. The correlations for the hopcount also reached a moderately high level because for sufficiently small α, all shortest paths in the graph lie on a single spanning tree [12]. As a consequence, the triangle inequalities are satisfied not only for path weights but also for the hopcount.

In conclusion, the hopcount on minimal weight paths satisfies the triangle inequalities both in the limit of weak and strong link weight fluctuations. This result is actually quite robust and independent from the graph topology and details of the link weight distribution. In the Internet path selection is obviously more complicated since routing policies play an important role. It remains an interesting question to model the effect of these policies on network distance matrices.

7.3.4 Observation of the First Experiment

We evaluated a landmark-based estimation scheme using real data for the delay and hopcount between Internet hosts, particularly, studying which hyperspace distance function correlates best with the actual distances between hosts. From our experiments and simulations we could draw the following conclusions:

- The best results for the delay estimation are obtained using D_+.
- The best results for the hopcount estimation are obtained using D_A.
- Hopcount is generally harder to estimate than delay.
- Theoretically, we found that even if a network is not embedded in a Euclidean space, and routes are based on link weight minimization, the hopcount distance matrix can be sufficiently correlated to support landmark-based estimation.

7.4 Experiment of the Aging of Landmark-based Coordinates

The second experiment is to investigate the aging of landmark-based coordinates, in particular, it is to investigate the quality of prediction based on coordinates which have been determined earlier. We analyzed the data collected by TTM from February 8, 2004 to February 14, 2004, and the data collected on May 15, 2003 and on January 31, 2004. We only considered the minimal delays available in all the days, resulting in 73 active boxes, where 62 hosts are located in Europe, 7 in the US, 2 in Japan, and 1 each in Australia and New Zealand. We omitted pairs for which the delay in one of the directions is missing, leaving a total of 4521 pairs. These measurements provide us with the network distance matrix $d(A, B)$.

In the experiment, over more than a week, 10 of the test-boxes were assigned the status of landmark. We choose them randomly except that 6 are located in the EU, 1 in Asia and 3 in the US. We repeated the experiments with $M = 5$ and $M = 15$ landmarks respectively. In the following Section 7.4.1, we investigate the quality of a landmark scheme over one week, while in the Section 7.4.2, we study the time dependence of coordinates.

7.4.1 The Quality of a Landmark Scheme over One Week

This investigates the quality of a landmark scheme over a week. We consider the measurement data that spans a week from Monday (February 8, 2004) to Sunday (February 14, 2004). On each day of the experiment, for each pair of peers, the current network distance $d(A, B)$ was calculated and compared to the distance predicted by the landmark scheme (using 5, 10 and 15 landmarks); *i.e.* the hyperspace distance $D(\boldsymbol{x}_A, \boldsymbol{x}_B)$ for six different hyperspace norms: D_1, D_2, D_∞, D_+, D_A, and D_G. Figure 7.3 plots the correlation coefficients $\rho(t, 0)$ for the estimated (i.e. hyperspace) distance versus the measured distance at the same day, for the six hyperspace distance functions. The vertical axis gives the linear correlation coefficient $\rho(t, 0)$, the horizontal axis the day.

Note that due to the dynamic property of the Internet distance, $\rho(t, 0)$ for six hyperspace distance functions are not always the same during a week. The results in Figure 7.3 show that during a week, the best results for the delay estimation are always obtained using D_G and D_+, and they seem to possess a relatively stable correlation over time. The difference between the results obtained using D_G and D_+ is within only 2%. Moreover, the D_G and D_+ hyperspace distances give the highest correlations, in particular higher than for the Euclidean distance D_2. This analysis confirms the conclusion of earlier experimental results, which is based on data measured 9 months earlier (May 15, 2003).

7.4.2 Estimate the Distance Using the Past Data

Hosts do not continuously update their coordinates. Here, we investigate the quality of prediction based on coordinates determined earlier. The central question addressed in here is: how often should peers update their coordinates? To be more precise, with a time lag between embedding and evaluation, how does the hyperspace distance function correlate with the actual distances between hosts?

To answer above questions, we took the distance measured on February 14, 2004 as measured distance $d(A, B)$. We predicted the distances based on data measured on earlier time (for six hyperspace distance functions). For each peer, the correlation coefficients $\rho(t, \Delta t)$ for the estimated (*i.e.* hyperspace) distance versus $d(A, B)$ was computed. Figure 7.4 shows an example of scatterplots for the measured distance versus the estimated (i.e. hyperspace) distance based on the data measured in 1 day earlier ($\Delta t = 1$). Each data point corresponds to a pair of peers. The horizontal axis shows the actual network distance $d(A, B)$ measured on February 14, the vertical axis the distance predicted by the landmark scheme, that is their hyperspace distance $D(\boldsymbol{x}_A, \boldsymbol{x}_B)$, using the previous delay data measured on February 13. The solid lines shown are the least-squares fitted line, and the diagonal $D = d$. The inset also shows the linear correlation coefficient $\rho(t, \Delta t)$.

Note that Internet distance can change due to routing policy, routing updates or

changes of the topology. Figure 7.5 presents the scatterplots of the distance measured on February 14 and some earlier measured distances. The results show that most end-to-end distances are stable in our database. In a number of cases, higher capacity links were chosen due to routing updates between source-destination pairs, which led to shorter end-to-end delays. Some outliers in Figure 7.5 show this change. For example, the routings from a host located in Sofia, Bulgaria to other test-boxes were completely changed since February 2004, and the one-way delay has been reduced by a factor of 10.

For a better visualization, Figure 7.6 shows the correlation coefficient $\rho(t, \Delta t)$ as a function of the time lag Δt. This experiment involves $M = 10$ landmarks. The plot shows that, based on the data measured up to a week earlier, the predicted hyperspace-distance function using D_G and D_+ correlates strongly (more than 0.85) with the measured distance and, specifically, exceeds that of the Euclidean distance D_2. The $\rho(t, \Delta t)$ is much lower (about 0.2) for the prediction based on data measured 14 days or more days earlier. Similar results were observed for $M = 5$ and $M = 15$.

	4 days earlier	**3 days earlier**	**2 days earlier**	**1 day earlier**
D_1	0.79, 0.85, 0.84	0.81, 0.87, 0.86	0.87, 0.91, 0.9	0.9, 0.89, 0.89
D_2	0.81, 0.85, 0.84	0.82, 0.86, 0.85	0.88, 0.9, 0.89	0.9, 0.9, 0.9
D_∞	0.81, 0.75, 0.74	0.82, 0.84, 0.82	0.86, 0.87, 0.86	0.88, 0.87, 0.87
D_+	0.94, 0.94, 0.93	0.93, 0.93, 0.91	0.97, 0.94, 0.95	0.97, 0.94, 0.93
D_A	0.9, 0.87, 0.86	0.89, 0.87, 0.85	0.94, 0.9, 0.89	0.95, 0.93, 0.93
D_G	0.92, 0.91, 0.91	0.92, 0.91, 0.9	0.96, 0.95, 0.94	0.97, 0.95, 0.95
	9 months earlier	**14 days earlier**	**6 days earlier**	**5 days earlier**
D_1	0.34, 0.38, 0.4	0.41, 0.48, 0.40	0.73, 0.79, 0.78	0.75. 0.82, 0.81
D_2	0.38, 0.41, 0.43	0.46, 0.52, 0.44	0.74, 0.79, 0.78	0.77, 0.82, 0.81
D_∞	0.48, 0.47, 0.49	0.56, 0.56, 0.51	0.76, 0.82, 0.8	0.79, 0.83, 0.82
D_+	0.69, 0.68, 0.68	0.74, 0.74, 0.66	0.87, 0.87, 0.86	0.93, 0.93, 0.91
D_A	0.58, 0.57, 0.59	0.65, 0.66, 0.6	0.84, 0.82, 0.82	0.88, 0.86, 0.85
D_G	0.58, 0.58, 0.59	0.66, 0.66, 0.6	0.85, 0.85, 0.85	0.9, 0.9, 0.89

Table 7.4: Correlation coefficients for the random graph simulations.

The complete results are summarized in Table 7.4, where the first, second and third number in each cell is the linear correlation coefficient $\rho(t, \Delta t)$ for $M = 5$, 10 and 15 landmarks, respectively. Table 7.4 confirms the observation that, based on the data measured within a week, the predicted hyperspace-distance function using D_G and D_+ correlate strongly with the measured distance.

To measure how well a predicted distance (based on data measured on earlier time) matches the corresponding measured distance, we use two metric relative errors defined as: $e(D_G, d) = |\frac{D_G - d(A,B)}{\min(D_G, d)}|$ and $e(D_+, d) = |\frac{D_+ - d(A,B)}{\min(D_+, d)}|$.

Figure 7.7 shows the cumulative distribution function (CDF) of the ratios of $e(D_G, d)$ and $e(D_+, d)$ for $M = 10$ for all measurement in our data sets. The closer this ratio is to 0, the better the estimation. We observed that, based on the data within a week earlier, between 94% and 98% of estimates using D_+ fall within a factor of 1 of the real distances; while 90% and 94% for D_G. However, based on the data measured 9 months earlier, only about 80% of estimates using D_+ (or D_G) fall within a factor of 1 of the real distances. A prediction mechanism can potentially be extremely inaccurate with respect to the relative error metrics, but the prediction scheme using D_G and D_+ based on the past data (*e.g.* within a week) can be relatively accurate compared to the measured distance. Similar results were also observed for $M = 5$ and $M = 15$.

7.5 Related Work

A closely related work is the GeoPing approach. GeoPing was originally intended to infer geographical locations of nodes from delay measurements to a set of landmarks [72], but was later also used to infer the delay itself [58]. In the latter form it is similar to the method studied by us, except that they only use the Euclidean hyperspace distance. A main conclusion of our work is that for predicting the delay, the Euclidean distance function is not the optimal choice. A minor difference is that they use the median delay in their later paper, while we use the minimum delay. However, none of them investigated the aging effect of landmark-based coordinates.

Guyton and Schwartz [42] studied the problem of selecting the nearest Internet servers in terms of hopcount. They compare various methods including (among several methods requiring router-support) the landmark-based scheme. They use the "triangulation" hyperspace distance function D_A first proposed in [44]. The focus in [42] is on the cost of the methods in terms of the number of packets exchanged. The effectiveness of "triangulation" is also addressed as well, but using a different measure than we do. We do not consider costs and focus on the landmark-method, for which we evaluate the effectiveness using various hyperspace distance functions. Our analysis confirms that D_A is a satisfactory choice for the hopcount. An additional benefit of our experiment is that we can compare the effectiveness of hopcount and delay estimation on the same set of nodes.

Global Network Positioning (GNP) by Ng and Zhang [69] is a different estimation scheme also based on landmarks. In this approach not just peers but also the landmarks are embedded in the hyperspace, based on the measured inter-landmark distances. The assignment of coordinates to landmarks and other nodes is cast in the form of a minimization problem where the objective function quantifies the errors in the distance estimates. This minimization need to be solved online, imposing a computational load on nodes and landmarks. The authors provide an extensive comparison between GNP, the "triangulation heuristics" D_+, D_∞, and D_A that we consider in this thesis, and

IDMaps (see below). They concluded that D_+ is the best triangulation heuristic for the delay. This is confirmed by our analysis. GNP is more accurate, at the cost of a higher computation and communication complexity. Lighthouse [81] and SCoLE [96] improve on GNP in the sense that the role of the landmarks is progressively decentralized.

IDMaps [37] is an infrastructure for estimating distances. Their "tracers" are servers similar to our landmarks, except that they actively measure their distances towards other tracers and towards domains on the Internet.

Vivaldi [26] is a scheme that assigns coordinate space for each host, but it does not require any fixed landmark. In Vivaldi, each node computes coordinates for itself. Vivaldi chooses coordinates by sampling the network latency between each node and a few other nodes, and adjusting the nodes' coordinates to minimize the error between the predicted and sampled latencies. Although Vivaldi is fully distributed, it costs time to sample all nodes at relatively same rate to ensure accuracy, and packets need to add Vivaldi-specific fields.

Various overlay networks use distance estimates to construct efficient topologies. Ref. [85] describes the use of a landmark scheme in a distributed hash-table called CAN.

7.6 Chapter Summary

In this chapter, we studied which hyperspace distance function correlates best with the actual distances between hosts. Our experimental results show that the best results for the delay and hopcount estimation are not always obtained using Euclidean distance. And Hopcount is generally harder to estimate than delay. We also studied how often peers should update their coordinates. More specifically, we investigated the quality of landmark schemes over one week using empirical delay data. Our experiments show that the best results for the delay estimation are obtained using D_G and D_+, which seem to possess a relatively stable correlation over time. The D_G and D_+ hyperspace distances give the highest correlations, and exceed that of the Euclidean distance D_2. We also investigated the quality of prediction based on coordinates determined earlier. Our experimental results suggest that based on data up to 7 days old, the predicted hyperspace-distance function using D_G and D_+ correlate relative highly with the measured distance. Moreover, those predicted hyperspace distance can be relatively accurate with respect to the measured distance.

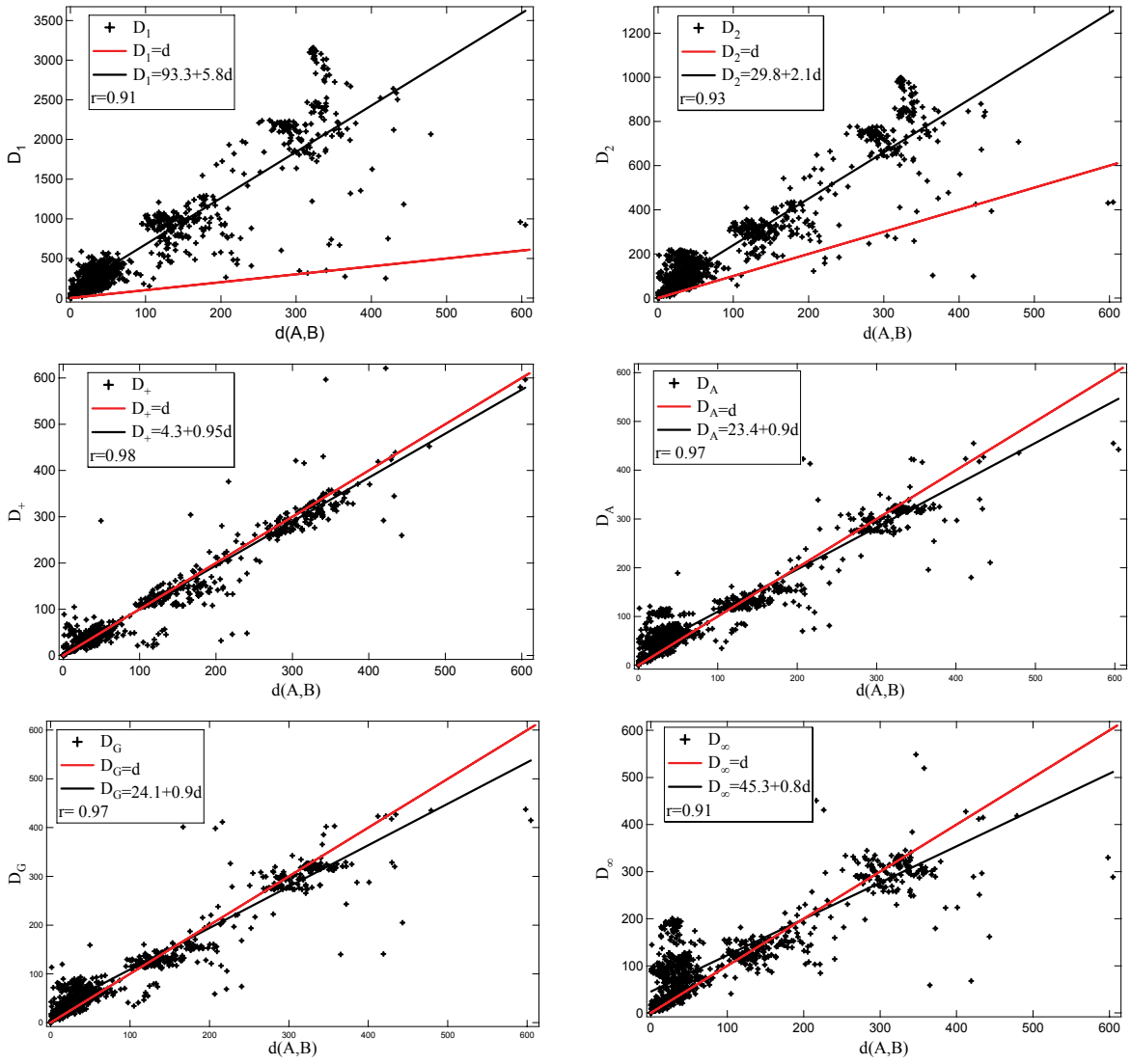


Figure 7.1: Scatterplots for the delay (round-trip) measurements. Shown is the estimated (i.e. hyperspace) distance versus the measured distance, for six hyperspace distance functions. The number of landmarks is 10. The plotted lines are the diagonal, $D = d$, and a least-squares fit.

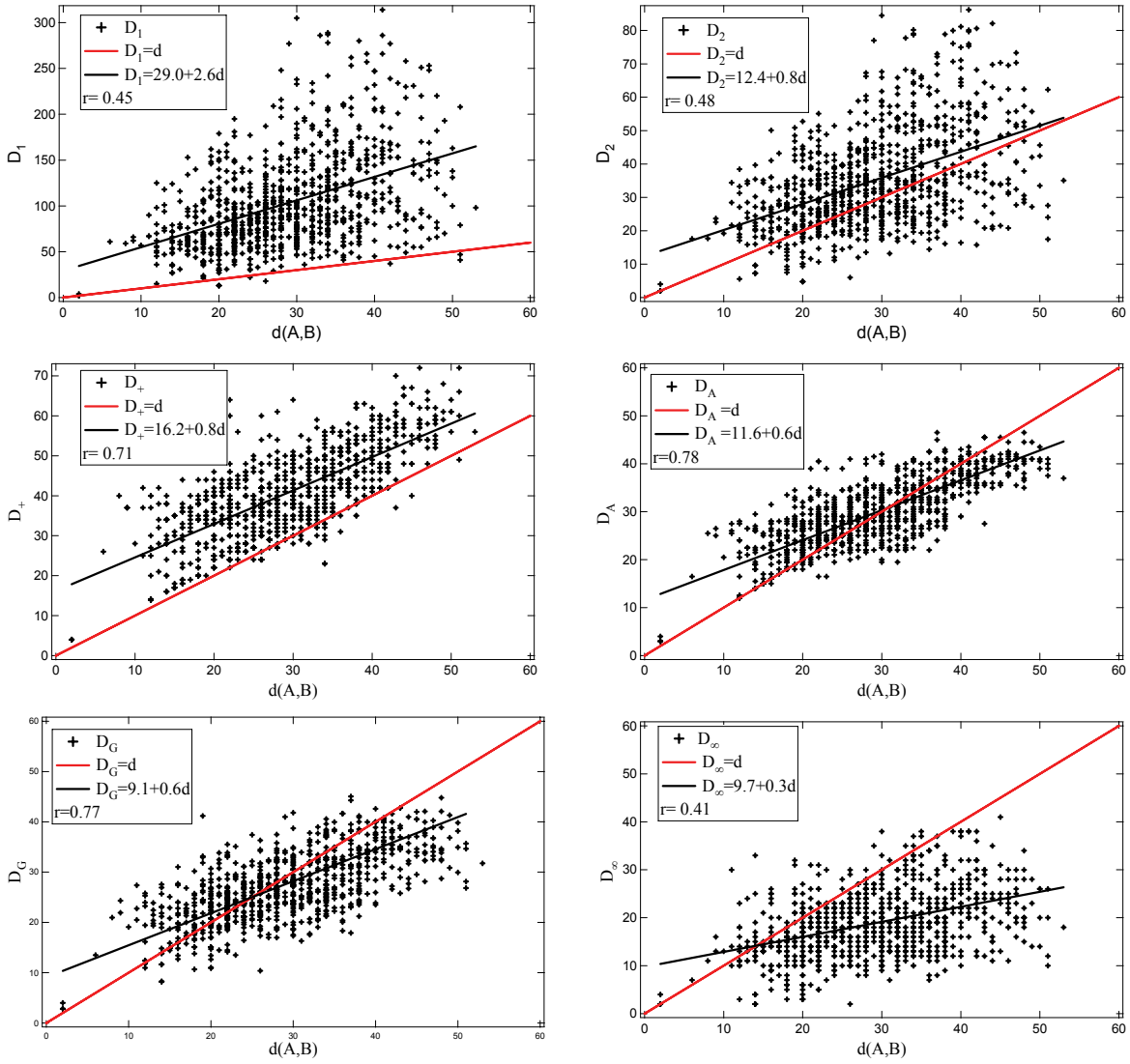


Figure 7.2: Scatterplots for the hopcount (round-trip). The number of landmarks is 10.

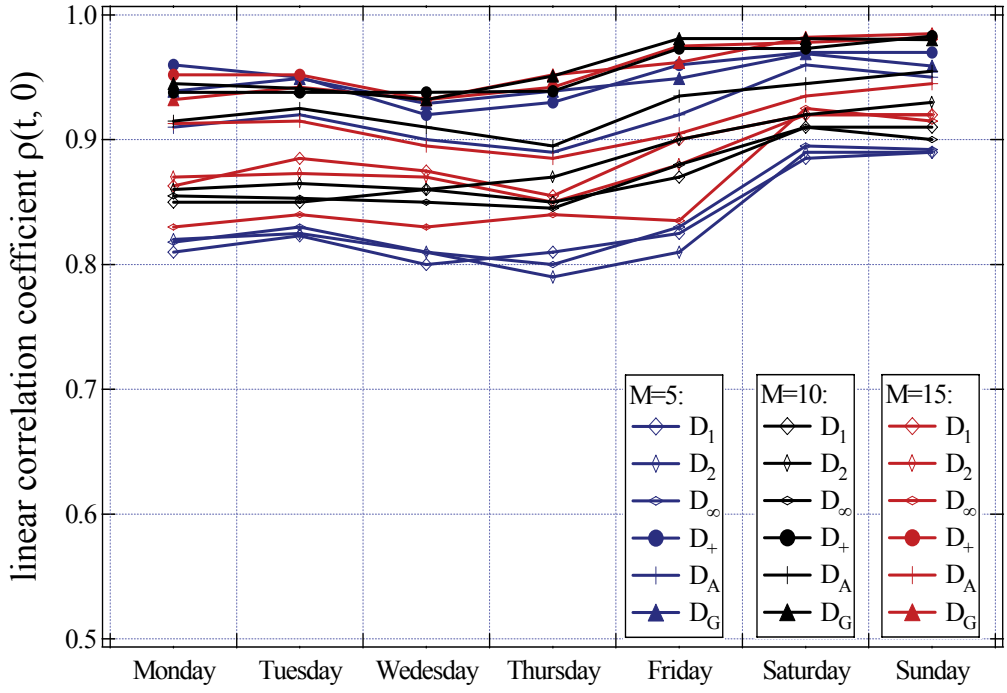


Figure 7.3: The correlation coefficients for the estimated (i.e. hyperspace) distance versus the measured distance, for six hyperspace distance functions.

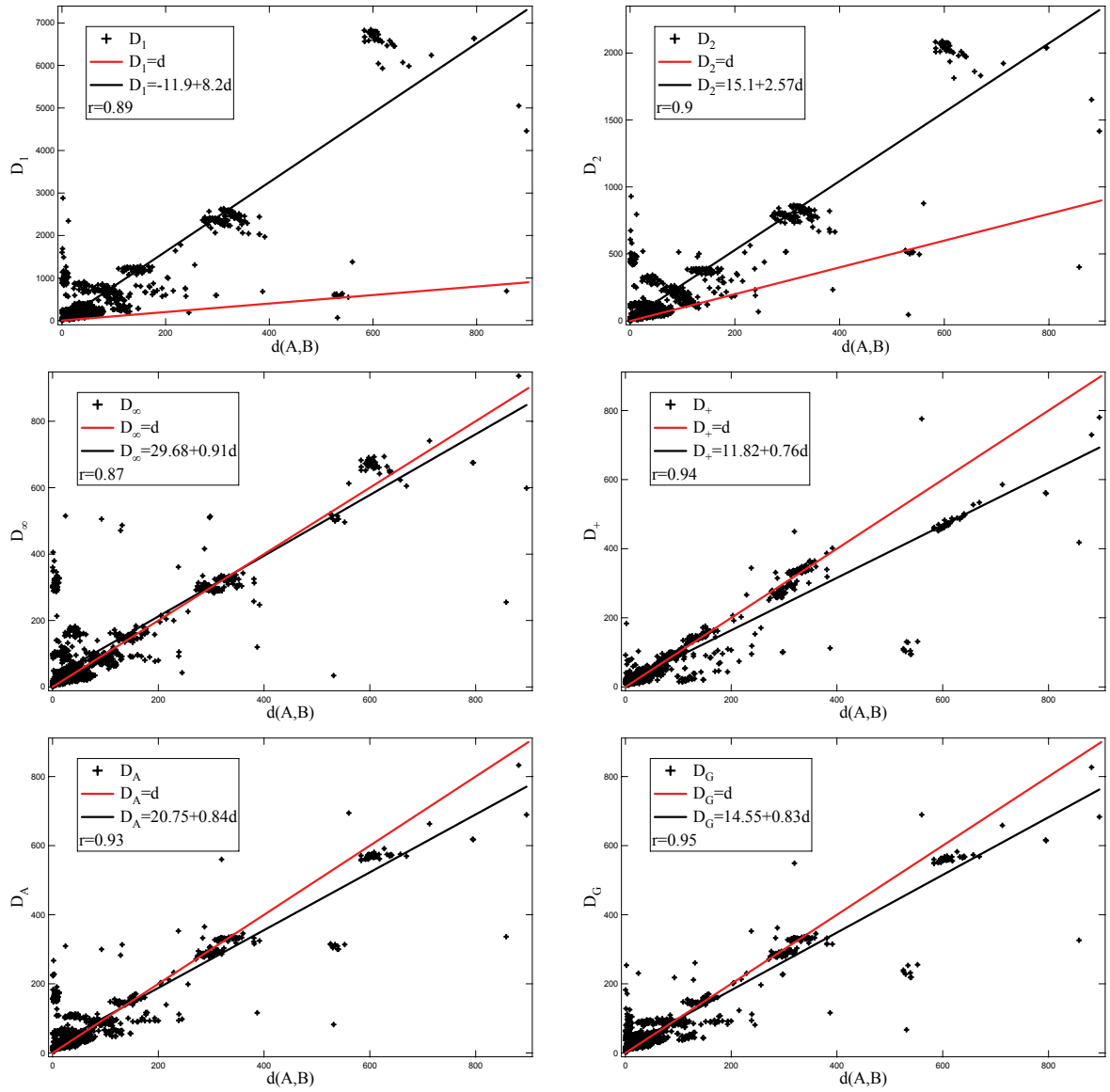


Figure 7.4: Scatterplots for the estimated (i.e. hyperspace) distance (using data from 1 day eariler) versus the measured distance, for six hyperspace distance functions. The number of landmarks is 10. The plotted lines are the diagonal D = d, and a least-squares fit

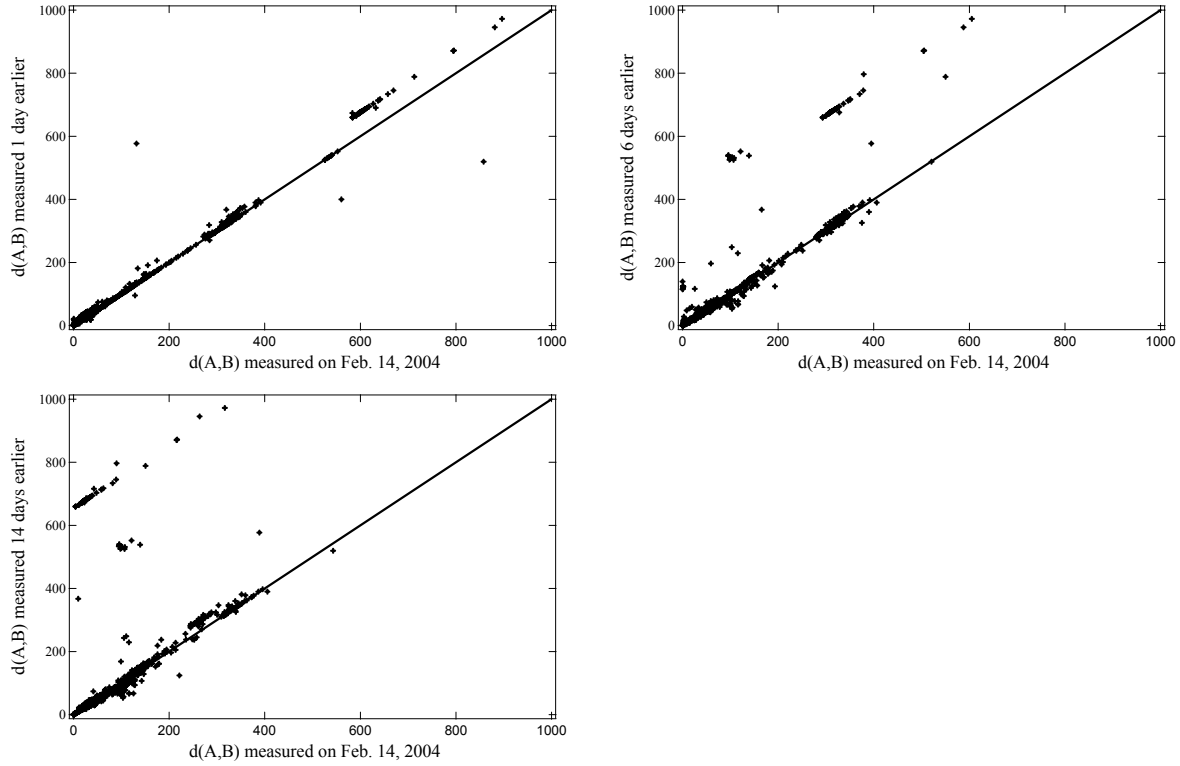


Figure 7.5: Scatterplots current distance (measured on Feb. 14, 2004) versus previous measured distance in the database. The plotted line is the diagonal

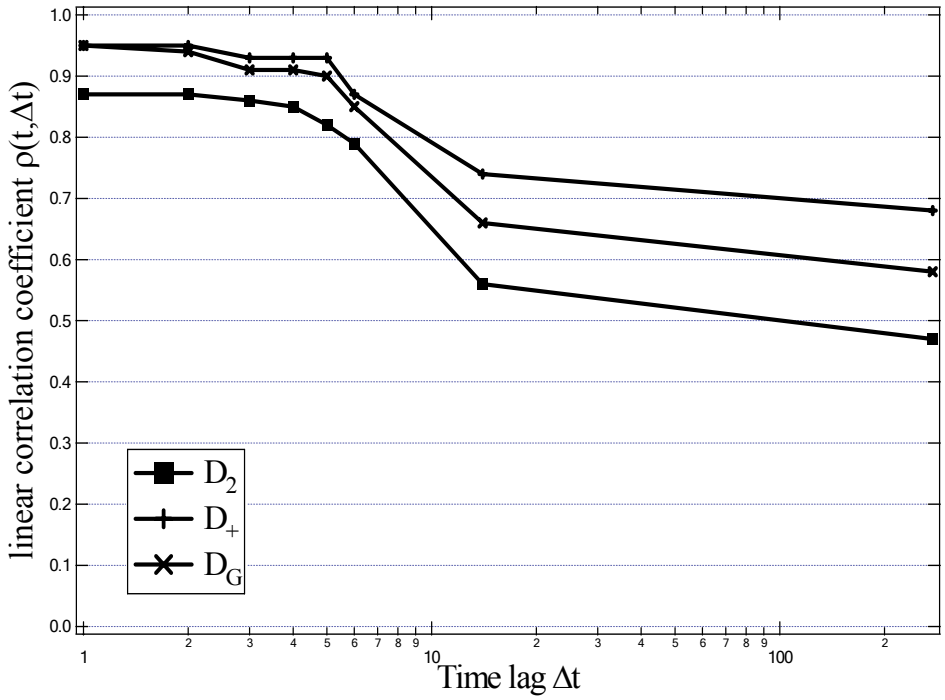


Figure 7.6: The correlation coefficient for the measured distance versus the estimated hyperspace distance based on data measured on eariler time for D_2, D_G and D_+. The landmarks is 10.

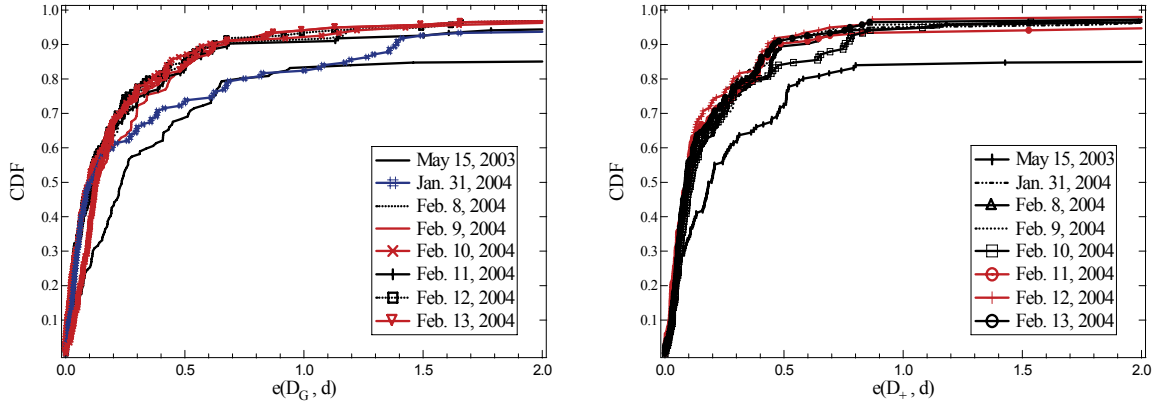


Figure 7.7: Cumulative distribution function (CDF) of the ratios of $e(D_G, d)$ and $e(D_+, d)$ for 10 landmarks in our experiment

Chapter 8

Conclusions

This thesis presents what is known about the field of Internet measurements and provides more insight into the field. It presents what we have learned about the measurement challenges, the measurement methods, and the measurement results. We focused on a set of standard metrics that can be applied to measure the quality, performance, and reliability of Internet data delivery services. These metrics include connectivity, one-way packet delay, loss, and reordering. The first part has therefore contributed to the following network-layer measurements:

- hopcount distribution and node degree distribution
- packet reordering
- IPv6 delay and loss evolution

For the end-users, it is not the network performance that matters most but the perceived quality of the applications running over the network. In general it can be said that the large scale deployment of applications will only be successful if the perceived quality of these applications is sufficiently high. We have also investigated how end-user perception of application quality is influenced by network lower-layer performances. The second part, therefore, contributes to the following application-layer measurements:

- VoIP perceived quality
- Peer-to-peer location estimation

The measurements and results obtained from each part above are summarized as follows:

Hopcount distribution and node degree distribution

Even though both the hopcount distribution and node degree distribution may provide greater insight in some important global characteristics of the Internet structure, reliable characterizations in IP graph are not currently available. The main goal of this section was to provide greater insight into possible hopcount and degree distributions of the IP graph, based on different datasets. In our study, the *traceroute* tool was used to collect the path information between any source-destination pair. We analyzed more than 24 million traceroutes collected by RIPE TTM, about 10 million traceroutes collected by the CAIDA skitter project, and thousands of traceroutes collected in a large testbed named PlanetLab. For quality analysis, first, traceroute routing pathologies have been analyzed and excluded from further study. Second, representative graphs have been constructed showing the union of most frequently occurring paths in traceroute measurements for each dataset. Our measurement results show that the hopcount distribution in the Internet (the distribution of path lengths in hops) can be well modeled by that of a random graph with uniformly or exponentially link weights. It should be noted that the node degree distributions may vary with different measurement sizes: our results show that for large group sizes, the node degree distribution appears to obey a power-law, while for small group sizes, the node degree distribution appears better fitted with exponential distribution.

Packet reordering

Reordering can occur on the high-speed links. The major cause of reordering has been found to be parallelism in Internet components (switches) and links. Reordering may also be caused by hardware configuration (*i.e.*, multiple switches in a router) and software (*i.e.*, class-based scheduling or priority queueing) in the routers. Reordering significantly impacts on the performance of applications in the Internet in terms of reducing throughput.

To understand the nature of reordering and to examine to what extent that reordering can influence the Internet delay performance, we analyzed the measurements of the end-to-end packet reordering by tracing UDP packets between 12 testboxes in RIPE NCC in two experiments. First, between each sender-destination pair of measurement boxes, IP probe-streams of a back-to-back burst of 50 100-byte UDP packets, called probe-streams, were continuously transmitted with interarrival times of about 30 seconds, resulting in a total of about 360 probe-streams in 3 hours from 5 to 8 PM (Greenwich Mean Time) on October 16, 2003. Second, we repeated the same experiment with a burst of 100 UDP packets in 3 hours from 5 to 8 PM on October 17, 2003. In order to obtain the snapshot of traffic patterns in the Internet, we limited each measurement to a total of 3 hours. We tagged the first experiment by N_{50} and the second by N_{100}. Our results lead to several observations:

1). Reordering depends on the network load, although below a certain load very little reordering occurs. In N_{50} , about 56% of the probe-streams had at least one reordering event, and about 66% in N_{100}.

2). Most individual streams have a relatively small number of reordering events. In N_{50} , about 14% of probe-streams had more than two reordering events, and about 26% in N_{100}.

3). A reordered stream is defined as a stream with at least one reordered packet received at the destination. Around 60% of reordered streams contain more than 4 reordered packets in N_{50}, and around 65% in N_{100}.These definitely impact a UDP performance because the reordering adds a high recovery cost to the end host with a finite buffer. On the other hand, packet reordering does not have a significant impact on the UDP delay (in N_{50} , 90% reordered packets had only a 5% greater delay to reach the destination).

4). The large interarrival time (30 seconds) between streams appears to be long enough to treat the streams as independent.

5). The asymmetry of reordered streams ratios exist on all experiment paths, but it varies greatly from testbox-to-testbox. For example, depending on the measurement set, the percentages of reordering of two directions in a random path can be the same, or upto 2.3 times different.

IPv6 Delay and Loss Performance Evolution

IPv6 is currently moving towards commercial deployment, and knowledge of its performance is crucial for ISPs to understand how to provide a high quality IPv6 network for future Internet applications. For IPv6 packet end-to-end delay and loss performance evolution, the measurement study was based on the RIPE infrastructure. By comparing IPv6 and IPv4 performance on a path-by-path basis, we have shown that the average end-to-end delay of IPv6 is about 61% higher than IPv4, and that packet losses are small (less than 0.02%) for both IPv6 and IPv4. Observing the evolution of IPv6 over time, we found that since October 2003, the median IPv6 delay has decreased from 120 ms in Oct. 2003 to about 55 ms in Oct. 2005. Packet loss for both IPv6 and IPv4 has remained small over the two year period.

IPv4 and IPv6 are expected to coexist for a while, and IPv6 tunnels are likely to play a key role in the transition phase. Software-based routers and tunnels provide a quick way to deploy IPv6. The drawback of this is that IPv6 tunneling degrade the traffic performance, mainly in terms of longer delay. On average, the median delay of IPv6 tunneling is about 87% higher than that of their IPv4 counterparts. Clearly, our results suggest that for a better IPv6 quality, only native IPv6 and hardware-based routers should be used everywhere. Still, the performance quality of software-based routers and tunnels is still acceptable for a successful transition phase.

VoIP perceived quality

Voice over IP (VoIP) is an increasingly popular and a cheap alternative to public switched telephone networks (PSTNs). VoIP technology enables voice traffic transferred over the Internet, allows easy introduction of new multimedia services, and supports

more flexibility in terms of codecs. For example, PSTNs are bound to a single codec G.711, while VoIP can use any codec as long as it is supported by the user applications. However, due to the connectionless, packet-switched characteristics of IP networks, packets may experience different delay, delay variation, reordering, and loss.

To understand how above factors (*i.e.* different delay, packet reordering, and packet loss) affect the perceived quality of voice calls, we have analyzed the measurements of end-to-end VoIP packets by tracing UDP packets between 12 testboxes in the Netherlands. We show that voice probes experience low delay in the network. Around 95% of all experimental source-destination pairs have a 99 percentile delay less than 114 ms. We also show that voice probes experience low loss in the network. Most (above 70%) individual source-destination pairs for different codecs have no packet loss, while about 99.5% of all the source-destination pairs for different codecs have less than 1% packet loss. Furthermore, our experiments show that the reordering of packets has no impact on the voice quality.

To determine the quality of VoIP calls over time, we have monitored the end-to-end packet delay and packet loss over 10 days. The VoIP packets were encoded with six different codecs. Our measurements suggest that the paths with low delay and loss can achieve an excellent Mean Opinion Score (MOS) value ($4 \leq MOS < 4.4$) at all times except for the rare cases when outages occur (*i.e.* system updates in the testboxes). The experimental results indicate that the networks in the Netherlands can continuously achieve a high VoIP quality.

Peer-to-peer location estimation

An efficient mechanism to estimate distance in the Internet would be useful for many large-scale distributed network applications such as nearby server selection and peer-to-peer computing, for example, when a client needs to select the nearest from a set of equivalent servers. Similarly, optimization of overlay networks (like peer-to-peer networks) often requires nodes connect to peers in their neighborhood. We have studied the hopcount and delay. The key question in this section is "Is there any method of estimating network distance based on reduced or incomplete measurements (*e.g.*, the delay and the hopcount)?"

To address this question, we evaluated a landmark-based estimation scheme using real data for the delay and hopcount between Internet hosts. Our large scale measurements show the method to be accurate and scalable. We further studied which hyperspace distance function (*e.g.*, Euclidean distance, the lower and upper bounds for the network distance) correlates best with the actual distances between hosts and show that the best results for the delay estimation are not always obtained using Euclidean distance.

Our measurement results indicate that hopcount is generally harder to estimate than delay. Using hopcount, the estimated network distance correlates worse (at least 20% worse) with the actual distances.

We have also studied how often peers should update their coordinates. Our experimental results suggest that when based on data up to 7 days old, the predicted hyperspace-distance functions correlate relative highly (73%) with the measured distance. In other words, predicted hyperspace distance can be relatively accurate compared to the measured distance.

Chapter 9

Abbreviations

AS	Autonomous System
BGP	Border Gateway Protocol
CAIDA	Cooperative Association for Internet Data Analysis
CCDF	Complementary Cumulative Distribution Function
CDF	Cumulative Distribution Function
GPS	Global Position System
GRE	Generic Routing Encapsulation
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPPM	IP Performance Metrics
ISP	Internet Service Provider
ITU	International Telecommunication Union
MOS	Mean Opinion Score
MTU	Maximum Transmission Unit
NIC	Network Interface Card
NTP	Network Time Protocol
OS	Operating System
OSI	Open Systems Interconnection
P2P	Peer to Peer
pdf	Probability density function
QoS	Quality of Service
RFC	Request for Comments
RIPE	Réseaux IP Européens
RTT	Round trip time
TCP	Transmission Control Protocol
TTL	Time to Live
TTM	Test Traffic Measurement
UDP	User Datagram Protocol
VoIP	Voice over IP

Bibliography

[1] Abramovitz, M., and Stegun, I. A. *Handbook on Mathematical Functions.* Dover Publications, 1968.

[2] Adam, Y., Fillinger, B., Astic, I., Lahmadi, A., and Brigant, P. Deployment and test of IPv6 services in the VTHD network. *IEEE Communications Magazine 42*, 1 (January 2004).

[3] Adamic, L. The small world web. *Proceedings of ECDL* (1999), 443–452.

[4] Allman, M., and Blanton, E. On making TCP more robust to packet reordering. *ACM Computer Communication Review 32* (January 2002).

[5] Almes, G., Kalidindi, S., and Zekauskas, M. RFC 2679: A one-way delay metric for IPPM. *IETF* (September 1999).

[6] Andersen, D. G., Balakrishnan, H., Kaashoek, M. F., and Morris, R. Resilient overlay networks. *Proc. of Symposium on Operating Systems Principles* (2001), 131–145.

[7] Beirlant, J., Teugels, J. L., and Vynckier, P. *Practical analysis of extreme values*, vol. 4. Leuven University Press, 1996.

[8] Bennett, C. R., Patridge, C., and Shectman, N. Packet reordering is not pathological network behavior. *Trans. on Networking IEEE/ACM* (December 1999).

[9] Bilinskis, I., and Mikelsons, A. *Randomized Signal Processing.* Prentice Hall International, 1992.

[10] Bollobas, B. *Random Graphs*, second ed. Cambrigde University Press, 2001.

[11] Borella, M., Swider, D., Uludag, S., and Brewster, G. Internet packet loss: Measurement and implications for end-to-end QoS. *International Conference on Parallel Processing* (Aug. 1998).

[12] Braunstein, L. A., Buldyrev, S. V., Cohen, R., Havlin, S., and Stanley, H. E. Optimal paths in disordered complex networks. *Phys. Rev. Lett. 91, 168701* (2002).

[13] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. Graphs structures in the web. *Computer Networks 33* (June 2000), 309–320.

[14] Bu, T., and Towsley, D. On distinguishing between Internet power law topology generators. *IEEE INFOCOM* (2002).

[15] CAIDA skitter. http://www.caida.org/tools/measurement/skitter/.

[16] Carpenter, B., and Jung, C. RFC 2529: Transmission of IPv6 over IPv4 domains without explicit tunnels. *IETF* (March 1999).

[17] Carpenter, B., and Moore, K. RFC 3056: Connection of IPv6 domains via IPv4 clouds. *IETF* (February 2001).

[18] CERN ROOT. http://root.cern.ch/.

[19] Chen, Q., Chang, H., Govindan, R., Jamin, S., Shenker, S. J., and Willinger, W. The origin of power laws in internet topologies revisited. *IEEE INFOCOM* (2002).

[20] Cho, K., Luckie, M., and Huffaker, B. Identifying IPv6 network problems in the dual-stack world. *SIGCOMM'04 Network Troubleshooting Workshop* (September 2004).

[21] Clark, D., Partridge, C., Braden, R. T., Davie, B., Floyd, S., Jacobson, V., Katabi, D., Minshall, G., Ramakrishnan, K., Roscoe, T., Stoica, I., Wroclawski, J., and Zhang, L. Making the world (of communications) a different place. *Computer Communication Review 35*, 3 (2005), 91–96.

[22] Clauset, A., and Moore, C. Traceroute sampling makes random graphs appear to have power law degree distributions. *arXiv:cond-mat/0312674 3*, 8 (Feb. 2004).

[23] Cole, R. G., and Rosenbluth, J. H. Voice over IP performance monitoring. *SIGCOMM Computer Communication 31*, 2 (2001), 9–24.

[24] Colitti, L., Battista, G. D., and Patrignani, M. Discovering IPv6-in-IPv4 tunnels in the Internet. *Proceedings of IEEE/IFIP NOMS 2004* (April 2004).

[25] Conta, A., and Deering, S. RFC 2473: Generic packet tunneling in IPv6 specification. *IETF* (December 1998).

[26] Dabek, F., Cox, R., Kaashoek, M. F., and Morris, R. Vivaldi: a decentralized network coordinate system. *SIGCOMM 2004* (2004), 15 – 26.

[27] Dedinski, I., Meer, H. D., Han, L., Mathy, L., Pezaros, D., Sventek, J., and Xiaoying, Z. Cross-layer peer-to-peer traffic identification and optimisation based on active networking. *Proc. of the IWAN 2005* (November 21-23 2005).

[28] Deering, S., and Hinden, R. RFC 2460: Internet protocol, version 6 (ipv6) specification. *IETF* (November 21-23 1998).

[29] Demichelis, C., and Chimento, P. RFC 3393: IP packet delay variation metric for IP performance metrics (ippm). *IETF* (November 2002).

[30] Ding, L., and Goubran, R. Assessment of effects of packet loss on speech quality in VoIP. *Proc. of the 2nd IEEE HAVE 2003* (2003), 49–54.

[31] Ding, L., and Goubran, R. A. Speech quality prediction in VoIP using the extended e-model. *IEEE GLOBECOM* (December 2003).

[32] Donnet, B., Friedman, T., and Crovella, M. Improved algorithms for network topology discovery. *In Proc. of Passive and Active Measurement Workshop (PAM)* (March 2005).

[33] Erdos, P., and Renyi, A. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci. 5*, 17Ű61 (1960).

[34] Faloutsos, C., Faloutsos, P., and Faloutsos, M. On power-law relationships of the Internet topology. *Proc. ACM SIGCOMM* (September 1999), 251–262.

[35] Floyd, S., and Jacobson, V. The asynchronization of periodic routing messages. *IEEE/ACM Transacions on Networking 2*, 2 (April 1994), 122–136.

[36] Floyd, S., and Jacobson, V. The synchronization of periodic routing messages. *IEEE/ACM Transactions on Networking 2*, 2 (April 1994), 122–136.

[37] Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. IDMaps: A global internet host distance estimation service. *IEEE/ACM Transactions on Networking 9*, 525 (2001).

[38] Gilligan, R., and Nordmark, E. Transition mechanisms for ipv6 hosts and routers. *RFC 2893* (August 2000).

[39] GlobalIPsound. http://www.ilbcfreeware.org/.

[40] Govindan, R., and Tanmunarunkit, H. Heuristics for Internet map discovery. *Proc. of IEEE INFOCOM 2000* (March 2000), 1371–1380.

[41] Graham, I., Donnelly, S., Martin, S., Martens, J., and J.Cleary. Nonintrusive and accurate measurement of unidirectional delay and delay variation over the Internet. *INETŠ98 Online Proceedings* (July 1998).

[42] Guyton, J. D., and Schwartz, M. F. Locating nearby copies of replicated internet servers. *Proc. ACM Sigcomm* (1995).

[43] Hanks, S., Li, T., and Traina, P. RFC 1701: Generic routing encapsulation (GRE). *IETF* (October 1994).

[44] Hotz, S. M. Routing information organization to support scalable routing with heterogeneous path requirements. *PhD thesis, Dept. of Computer Science, University of Southern California* (1994).

[45] Huitema, C. Teredo: Tunneling ipv6 over udp through network address translations (nats). *RFC 4380* (February 2006).

[46] Internet Engineering Task Force. http://www.ietf.org/.

[47] Internet Software Consortium. www.isc.org.

[48] ITU-T P.833. Methodology for derivation of equipment impairment factors from subjective listening-only tests.

[49] ITU-T Recommendation G.107. The E-Model, a computational model for use in transmission planning.

[50] ITU-T Recommendation G.113 appendix I. Provisional planning values for the equipment impairment factor Ie and packet-loss robustness factor bpl.

[51] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., and Towsley, D. Measurement and classification of out-of-sequence packets in a Tier-1 IP backbone. *Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2002* (November 2002).

[52] Jaiswal, S., Rosenberg, A. L., and Towsley, D. Comparing the structure of power-law graphs and the Internet AS graph. *IEEE ICNP* (2004).

[53] Janic, M. *Multicast in Network and Application Layer, PhD Thesis.* Delft University of Technology, Delft, The Netherlands , October 2005.

[54] Janic, M., Kuipers, F. A., Zhou, X., and Van Mieghem, P. Implications for QoS provisioning based on traceroute measurements. *QofIS2002* (October 2002), 3–14.

[55] Karagiannis, T., Broido, A., Brownlee, N., kc claffy, and Faloutsos, M. Is P2P dying or just hiding? *Globecom 2004* (November-December 2004).

[56] Keys, K., Moore, D., Koga, R., Lagache, E., Tesch, M., and claffy, K. The architecture of the CoralReef: Internet traffic monitoring software suite. *PAM 2001* (2001).

[57] Lakhina, A., Byers, J., Crovella, M., and Xie, P. Sampling biases in IP topology measurements. *Proc. 22nd ACM SIGCOMM Conference* (2003).

[58] Lakshminarayanan, L., and Padmanabhan, V. N. Network performance of broadband hosts: Measurements and implications. *Microsoft Technical Report* (2003).

[59] Laor, M., and Gendel, L. The effect of packet reordering in a backbone link on application throughput. *IEEE Network* (September 2002).

[60] Lawrence, S., Giles, C. L., and Bollacker, K. Digital libraries and autonomous citation indexing. *IEEE Computer 32*, 6 (1999), 67–71.

[61] Li, L., Alderson, D., Willinger, W., and Doyle, J. A first-principles approach to understanding the Internet router-level topology. *ACM SIGCOMM* (2004).

[62] Liu, X., , Ravindran, K., Liu, B., and Loguinov, D. Single-hop probing asymptotics in available bandwidth estimation: A sample-path analysis. *ACM IMC* (October 2004).

[63] Mahdavi, J., and Paxson, V. RFC 2498: IPPM metrics for measuring connectivity. *IETF* (January 1999).

[64] Markopoulou, A. F., Tobagi, F., and Karam, M. Assessment of VoIP quality over Internet backbones. *Proceedings of IEEE Infocom* (June 2002).

[65] Marsh, I., Li, F., and Karlsson, G. Wide area measurements of Voice over IP quality. *QofIS 2003* (2003).

[66] Mills, D. L. RFC 1305: RFC 1305 - network time protocol (version 3) specification, implementation and analysis. *IETF* (March 1992).

[67] Moore, A. W., and Papagiannaki, K. Toward the accurate identification of network applications. *PAM05* (2005), 41–54.

[68] Morton, A., Ciavattone, L., Ramachandran, G., and Perser, J. draft-ietf-ippm-reordering-12.txt. *IETF, work in progress*.

[69] Ng, T. S. E., and Zhang, H. Towards global network positioning. *Proc. ACM Sigcomm Internet Measurement Workshop* (2001).

[70] Odlyzko, A. Internet pricing and the history of communications. *Computer Networks (Amsterdam, Netherlands: 1999) 36*, 5-6 (2001), 493–517.

[71] Ohzahata, S., Hagiwara, Y., Terada, M., and Kawashima, K. A traffic identification method and evaluations for a pure P2P application. *PAM05* (2005), 55–68.

[72] Padmanabhan, V. N., and Subramanian, L. An investigation of geographic mapping techniques for Internet hosts. *Proc. ACM Sigcomm* (2001).

[73] Pansiot, J. J., and Grad, D. On routes and multicast trees in the internet. *ACM SIGCOMM Computer Communication Review 28*, 1 (Jan 1998), 41–50.

[74] Pastor-Satorras, R., and Vespignani, A. *Evolution and Structure of the Internet: A Statistical Physics Approach*. Cambridge University Press, 2004.

[75] Paxson, V. Automated packet trace analysis of TCP implementations. *In Proceedings of the 1997 SIGCOMM Conference* (September 1997), 167–179.

[76] Paxson, V. End-to-end Internet packet dynamics. *Proc. ACM SIGCOMM* (September 1997), 139–152.

[77] Paxson, V. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking 5*, 5 (October 1997), 601–615.

[78] Paxson, V. RFC 2330: Framework for IP performance metrics. *IETF* (May 1998).

[79] Paxson, V., Almes, G., Mahdavi, J., and Mathis, M. Framework for IP performance metrics. *RFC 2330* (May 1998).

[80] Paxson, V., and Floyd, S. Why we don't know how to simulate the Internet. *Proc. 1997 Winter Simulation Conference* (Dec 1997).

[81] Pias, M., Crowcroft, J., Wilbur, S., Harris, T., and Bhatti, S. Lighthouses for scalable distributed location. *Proc. of the 2nd International Workshop on Peer-to-Peer Systems* (2003).

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

[83] Plissonneau, L., Costeux, J.-L., and Brown, P. Analysis of peer-to-peer traffic on ADSL. *PAM 2005* (March 31 to April 1 2005).

[84] Popoviciu, C., Levy-Abegnoli, E., and Grossetete, P. *Deploying IPv6 Networks*. Print ISBN-10: 1-58-705210-5. Cisco Press, February 10 2006.

[85] Ratnasamy, S., Handley, M., Karp, R., and Shenker, S. Topology-aware overlay construction and server selection. *Proc. IEEE Infocom* (2002).

[86] Rexford, J., Wang, J., Xiao, Z., and Zhang, Y. BGP routing stability of popular destinations. *Proc. of ACM SIGCOMM Internet Measurement Workshop 2002* (November 2002).

[87] RIPE Test Traffic Measurements. http://www.ripe.net/ttm.

[88] RIPE Whois database. http://www.ripe.net/db/index.html.

[89] Route Views. http://www.routeviews.org/.

[90] Savage, S., Collins, A., Hoffman, E., Snell, J., and Anderson, T. The end-to-end effects of Internet path selection. *Proc. of ACM SIGCOMM* (1999), 289–299.

[91] Smythe, R. T., and Mahmoud, H. M. A survey of recursive trees. *Theory of Probability and Mathematical Statistics*, 51 (1995), 1–27.

[92] Spring, N., Mahajan, R., and Anderson, T. Quantifying the causes of path inflation. *Proc. of ACM SIGCOMM* (2003).

[93] Srivastava, V., Wargo, C., and Lai, S. Aviation application over IPv6: performance issues. *IEEE Aerospace Conference 3* (6-13 March 2004), 1661–1670.

[94] Stevens, W. R. *TCP/IP Illustrated*, vol. 1. Addision-Wesley, Massachusetts, 1994.

[95] Surveyor. http://www.advanced.org/surveyor/.

[96] Szymaniak, M., Pierre, G., and van Steen, M. Scalable cooperative latency estimation. *Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS)* (July 2004), 367–376.

[97] Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S., and Willinger, W. Network topology generators: Degree-based vs. structural. *ACM SIGCOMM* (2002), 147–159.

[98] Telephone Transmission Quality Objective Measuring Apparatus. Artificial conversational speech, ITU-T recommendation P.59 .

[99] Templin, F., Gleeson, T., Talwar, M., and Thaler, D. Intra-site automatic tunnel addressing protocol (isatap). *RFC 4214* (October 2005).

[100] Tsirtsis, G., and Srisuresh, P. RFC 2766: Network Address Translation - Protocol Translation (NAT-PT). *IETF* (February 2000).

[101] Uhlig, S., Magnin, V., Bonaventure, O., Rapier, C., and Deri, L. Implications of the topological properties of Internet traffic on traffic engineering. *19th ACM Symposium on Applied Computing, Special Track on Computer Networks* (March 2004).

[102] Van Mieghem, P. *Performance Analysis of Communications Networks and Systems*, 1 ed. Cambridge University Press, 2006.

[103] Van Mieghem, P., Hooghiemstra, G., and van der Hofstad, R. W. Scaling law for the hopcount. *Delft University of Technology, report2000125* (2000).

[104] Varshney, U., Snow, A. P., McGivern, M., and Howard, C. Voice over IP. *Commun. ACM 45*, 1 (2002), 89–96.

[105] Wang, Y., Ye, S., and Li, X. Understanding current IPv6 performance: a measurement study. *10th IEEE Symposium on Computers and Communications* (27-30 June 2005).

[106] Watts, D., and Strogatz, S. Collective dynamics of small-world networks. *Nature 363*, 202-204 (1998).

[107] www.limewire.com. http://www.limewire.com/english/content/netsize.shtml.

[108] Zhou, X., Muller, F., Kooij, R. E., and Van Mieghem, P. Estimation of voice over IP quality in the Netherlands. *IPS-MoMe 2006* (February 2006).

[109] Zhou, X., and Van Mieghem, P. Reordering of IP packets in the Internet. *PAM2004* (April 2004), 237–246.

[110] Zhou, X., and Van Mieghem, P. Hopcound and E2E delay: IPv6 Versus IPv4. *PAM2005* (March 31 - April 01 2005).

Acknowledgements

Pursuing a Ph.D. within four years is an amazing journey. During this journey, I met many outstanding and interesting people. Here I would like to thank many people for their time, interest, and willingness to help me with the thesis.

The first person I would like to thank is my supervisor and promoter, Prof. Piet Van Mieghem. He gave me the opportunity to join NAS group. Thanks to his expert guidance, I was able to complete my Master thesis and Ph.D. thesis. I would also like to express my gratitude to him for his patience, support and encouragement with me.

I would like to thank Dr. Gerard Hooghiemstra for his patience with me and his understanding.

I would also like to thank all members of my Ph.D. committee for reading my manuscript and providing useful feedback.

During my Ph.D. I enjoyed the pleasant and cheerful working environment created by the colleagues at the NAS and WMC groups in the $19th$ floor in the EEMCS building. I would especially like to thank Mark Santcroos, Milena Janic, Fernando A. Kuipers, Stijn Van Lange, Ramin Hekmat, Rob E. Koij, Martin Jacobsson, Anthony Lo and Alex Slingerland. My thanks also go to Calloline Bertin for her useful support.

I would like to thank Henk Uijterwaal and Rene Wilhelm for providing me with RIPE TTM Internet measurement data.

This Ph.D. thesis is partially the result of the research conducted among several Universities. I had the pleasure to work with several outstanding researchers. I would like to especially thank Eng Keong Lua (University of Cambridge, United Kingdom), Prof. Jon Crowcroft (University of Cambridge, UK), Michael Kleis (Fraunhofer Institut FOKUS), and Michael Sminov (Fraunhofer Institut FOKUS) for their soothing guidance, their understanding and support.

Finally, I would like to thank my parents and my elder sister. Without their support, this work would never have been possible.

Curriculum Vitae

Personal details

Title, name	Ir. Xiaoming Zhou
Date and place of birth	September 19, 1977, Guangdong
Nationality	Chinese

Profile

Xiaoming Zhou received M.Sc. degree in Electrical Engineering in 2002 from Delft University of Technology in the Netherlands. For his master's thesis he studied Internet behavior and made a mathematical model of Internet structure in the IP level, and evaluated the model by the simulation as well as through real Internet measurement. From July 2002 to December 2002, he did research in Delft University of Technology on Internet delay analysis funded by a Dutch Government scholarship. He cooperated in this research with RIPE TTM. His Ph.D. work mainly focused on Internet measurement and its impact to the applications like VoIP and P2P. He served as a reviewer for many journals and conferences, among them Computer Networks, INFOCOM, P2P, QofIS, CoNEXT, and NEW2AN.

Publications

- Zhou, X., R. E. Kooij, H. Uijterwaal, and P. Van Mieghem, "Estimation of Perceived Quality of Service for Applications on IPv6 Network", ACM/IEEE International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks 2006, Malaga, Spain, pp. 74-81, October 2-6, 2006.
- Tang, H, M. Janic, and X. Zhou, "Hopcount in the NICE Application Layer Multicast Protocol", IEEE/SMC the multiconference on Computational Engineering in Systems Applications, Beijing, China, October 4-6, 2006

- Lua, E. K., X. Zhou, J. Crowcroft, and P. Van Mieghem, "Hierarchical Geometric Overlay Multicast Network", IEEE INFOCOM 2006 Poster and Demo Session Program, Barcelona, Spain, April 23-29, 2006
- Zhou, X., F. Muller, R. E. Kooij and P. Van Mieghem, "Estimation of Voice over IP Quality in the Netherlands", to appear in 4th IEEE International Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MoMe 2006), Salzburg, Austria, February 27-28, 2006
- Janic, M., N. Ineke, X. Zhou and P. Van Mieghem, "Hopcount in Application Layer Multicast Schemes", to appear in Proc. of 4th IEEE International Symposium on Network Computing and Applications (IEEE NCA05), Cambridge, MA, USA, July 27-29, 2005
- Kleis, M., E. K. Lua and X. Zhou, 2005, "Hierarchical Peer-to-Peer Networks using Lightweight SuperPeer Topologies", The 10th IEEE Symposium on Computers and Communications (ISCC 2005), Cartagena, Spain, June 27-30, 2005
- Kleis, M., E. K. Lua and X. Zhou, "A Case for Lightweight SuperPeer Topologies", "Kommunikation in Verteilten Systemen" (KiVS) Workshop, Germany, 28. Feb. to 2 March, 2005
- Zhou, X. and P. Van Mieghem, "Hopcound and End-to-End Delay: IPv6 Versus IPv4", Proceedings of Passive and Active Measurement (PAM2005), Boston, MA, USA, March 31 - April 01, 2005
- Zhou, X. and P. Van Mieghem, "On the Aging of Landmark-based Coordinates", Joint conference of 10th IEEE Asia-Pacific Conference on Communications (APCC2004) and 5th International Symposium on Multi-Dimensional Mobile Communications (MDMC2004) (APCC/MDMC'04), Beijing, China, pp. 347-350, August 29-September 1, 2004
- Kleis, M., and X. Zhou, "Placement schemes for Peer-to-Peer Networks based on Principles from Geometry", The Fourth IEEE International Conference on Peer-to-Peer Computing Use of Computers at the Edge of Networks (P2P, Grid, Clusters), Zurich, Switzerland, August 25-27, 2004
- Zhou, X. and P. Van Mieghem, "Reordering of IP Packets in the Internet", Proceedings of Passive and Active Measurement (PAM2004), Antibes Juan-les-Pins, France, pp. 237-246, April 19-20, 2004
- Van Langen, S., X. Zhou and P. Van Mieghem, "On the Estimation of Internet Disctances Using Landmarks", International IEEE Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN'04), St. Petersburg, Russia, Februari 2-6, 2004

• Janic, M., F. Kuipers, X. Zhou and P. Van Mieghem, "Implications for QoS provisioning based on traceroute measurements", Proceedings of 3nd International Workshop on Quality of Future Internet Services, QofIS2002, Zurich, Switzerland, pp. 3-14, October 16-18, 2002