TKN Telecommunication Networks Group

Technical University Berlin
Telecommunication Networks Group

Measurements of Path Characteristics in PlanetLab

Michael Olbrich, Felix Nadolni,
Filip Idzikowski, Hagen Woesner

mo-publish@freenet.de, uni@nadolni.org,
{idzikowski|woesner}@tkn.tu-berlin.de

Berlin, July 2009

TKN Technical Report TKN-09-005

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

This page has been intentionally left blank.

Abstract

We discuss findings from Internet measurements conducted within the global PlanetLab overlay network. Our study comprises results from two *traceroute* round-trip measurements, covering more than 250 nodes, as well as one-way packet dynamics measurements for another 20 selected nodes worldwide. For that purpose we devised a PCAP-based active probing framework with packet replay capabilites, using fixed-sized, Poisson sampled UDP probe packets. We combine our results with geotargeting information, and from that derive distance and Hop-Count dependent path characteristics, such as Delay, Delay Variation and Packet Loss. Based on assumptions provided by *ITU G.114*, packet delays are found to be averagely induced by one third within router queues and by two third due to signal propagation times. Spatial properties and routing behaviour within PlanetLab, including observed routing anomalies are treated as well. In order to assess measuring accuracy on PlanetLab nodes, we then examine the Linux timer system, show the impact of clock "noise" on the Poisson sampling process, and point out simple enhancements of the Linux kernel, that significantly improve timing accuracy. We show the impact of clock skew and provide theoretical background to encounter such uncertainties. Then real-world application performance is also adressed by utilizing Voice over IP measurements on particular *Internet2* paths, finding jitter not a limiting factor on voice quality.

Index Terms—Network performance, network monitoring, network topology, active probing, delay measurements, one-way metrics, clock dynamics, geolocation.

This page has been intentionally left blank.

Contents

List of Figures

List of Tables

1 Introduction

Since the early large-scale end-to-end Internet performance measurements by Paxson in 1997 [49] an ongoing effort was expended on the further development and enhancement of the Internet. But despite permanently growing line capacities, increasing routing hardware performances and more sophisticated flow control abilities such as by *Multi Protocol Label Switching* (MPLS), measurements of packet dynamics are still an essential resource in understanding and engineering *Wide Area Networks* (WANs). The widespread displacement of classical multimedia services such as telephony and television from circuit-switched to packet-switched networks has concurrently increased the demands on the performance of the Internet and therewith the needs in validating path characteristics of particular routes. While extensive larger-scale end-to-end measurements presented great logistical difficulties in the past, new world-wide distributed research platforms like PlanetLab offer great capabilities and gain access to a wide majority of paths in the Internet. Since many PlanetLab nodes are already connected to the *Internet2* [42], this further allows for the application development and performance estimation of a future Internet. A critical design issue therein would be the further reduction of router-induced queuing delay variability, to gain better support for multimedia services such as Voice over IP (VoIP) or video conferencing.

Towards a better understanding of packet dynamics in the Internet and their relation to the number of traversed routing points (hops), we devised a PCAP based end-to-end measurement framework to be used in PlanetLab. By utilizing synthetic UDP probe streams and following design and validation techniques specified by the IETF *IP Performance Metrics* Group (IPPM) [50], we obtained several one-way path characteristics on routes between a variety of world-wide distributed PlanetLab nodes. We combined our path characteristics measurements with geo-targeting techniques to further extend our cognitions by the knowledge of geographical positions and distances of particular hops. This allows in addition for the estimation of path lengths as well as geographical distribution and spacing of routing points within the Internet. In order to infer from synthetic measurements on real-world application performances and to investigate influences of the measurement hosts themselves, we also analyzed the quality of Voice over IP calls on particular paths and measured the clocking precision within PlanetLab nodes.

The remainder of the report is organized as follows. In Section 2, we give a brief introduction to IP performance measurements and present techniques and practical directions related to them. Section 3 describes the design of our measurement framework as well as the used evaluation criteria and methodologies. Finally, we present our measurement results in Section 4, including outcomes of geo-targeting measurements, one-way path characteristics and VoIP quality measurements, while Section 5 concludes the report.

2 Elements of IP Performance Measurements

To set the scene of this report, we start by giving a brief overview of techniques and metrics related to measuring and quantifying the performance of IP networks. We cover different approaches of monitoring packets, estimating link utilization and give some hints on common problems. In the last paragraph of this Section we also present some basic solutions to reduce such problems.

2.1 Active and Passive Measurements

Generally there exist two fundamental approaches to observe the behaviour of an unknown system. One approach is Active Measurements and the other is Passive Measurements. Each of them has its advantages and drawbacks. Mostly, either the one or the other is used for a specific task.

Active Measurements – are performed by injecting probe traffic into the network and measuring the bearing of test packets or responses to them. This method is suitable to monitor the performance of a network. Active Measurements can be performed anytime the observer wants to, he will always observe the current characteristics of the network, unless a majority of test packets gets lost. Furthermore, the active approach allows for explicit control on the generation of packets for different measurement scenarios. This includes control on the nature of traffic generation, the sampling techniques, the timing, frequency, scheduling, packet sizes and types (to emulate various applications), statistical quality, the path and function chosen to be monitored. The classical *Ping* program is an example for an Active Measurement application. Besides the mentioned merits, injected traffic can perturb the behaviour of the network and can lead to distortion of the measurement results though. For example, it can change the congestion level or drive the network into a state of synchronization. Also, test packets may be blocked by firewalls or will be treated at different priorities by routers. So the choice of an appropriate source model for probing packets is crucial to get optimal results. We will discuss this issue in more detail in Section 2.3.

Passive Measurements – are carried out by observing the network traffic and capturing packets from a link or a network flow at a specific router or node attached to the network. This method is suitable to analyze the usage and type of traffic that passes a network. Depending on the completeness of the captured data it is possible to get a rich detailed view on the network behaviour. Intrusion detection and firewall inspection systems are often based on Passive Measurements. However, the heavier the network traffic, the more packet capturing becomes a challenging task. Computing power and data storage require sophisticated hard and software, particularly if the specific application needs to operate in real-time. This can lead to performance degradation of the capturing device. Moreover, if capturing of all packets on the network is required, privacy or security can become an issue.

2.2 Network Monitoring Metrics

Now having shown approaches of *how* to measure we also need to define *what* to measure. Terms related to network measurements are sometimes used in different manners and their meanings in the particular context can become confusing. The definition of a clear terminonolgy is crucial to provide undisputed and reproducible measurement results. The two main representative standardization efforts on metrics related to network measurements are the *Cooperative Association for Internet Data Analysis* (CAIDA) and the *IETF IP Performance Metrics Group* (IPPM).

Within this work we will focus on the IPPM terminology as far as possible. The next Section gives a partial overview and classification of measurement metrics. In the subsequent Sections we will pick out and define the metrics used in this work in more detail.

2.2.1 Classification

Network measurements are aimed to answer most diverse questions. Depending on the type of application, from whose perspective, for which kind of protocol and so forth there are also many different measurement metrics. Some basic classes of measurement metrics are Availability, Loss, Delay, Utilization and Routing. A coarse classification of network monitoring metrics, derived from [34] and [45] is given in Fig. 1 and a short description follows below.

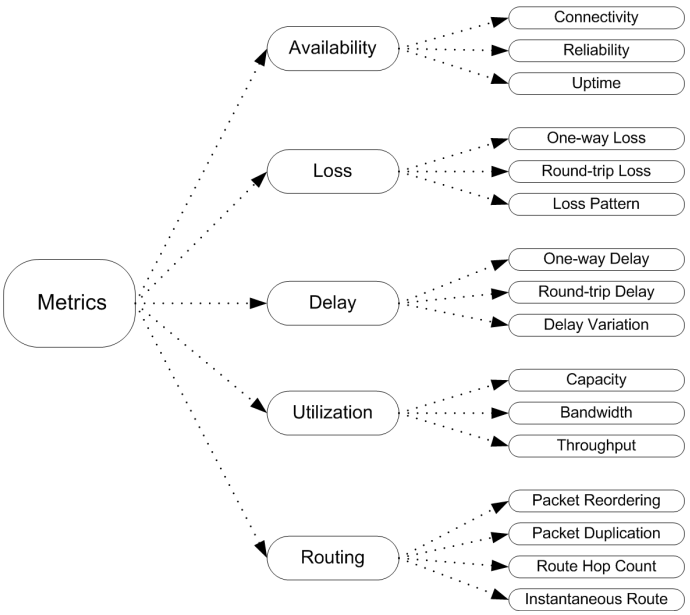


Figure 1: Network Monitoring Metrics.

Availability includes e.g. the percentage of a specified time a network or host is available for normal use.

Loss describes in general the fraction of packets that get lost during the transit from one host to another. Since the Internet mostly works on a best-effort basis routers can drop packets under some conditions.

Delay is the time taken for a packet to travel from one host to another. It can be further subdivided into different fractions of occurence of partial delays that sum up to the total delay, e.g. processing delay, serialization delay, propagation delay, queing delay and others.

Utilization reflects the amount of data sent through a given link or path in relation to its capacity. It may also denote the degree of memory usage of a router queue during a certain period in time.

Routing, at least, further denominates a wide range of metrics which comprise e.g. the instantaneous route of a network path, the hop count of a route or the frequency a route from one host to another changes.

Obviously the variety of network measurement metrics is magnifold. Also some metrics are derived from other metrics or classes of metrics. Beyond that many metrics must be seen conditioned by other metrics. It should also be stressed to distinguish between analytical and empirical specified metrics as noted by Paxson [45]. Analytical and empirical specified metrics differ in the manner that empirical metrics are mostly only a close approximation to the corresponding analytical ones conditional on the methodology used to quantify the metric.

To avoid ambiguities with metrics we use terminology and definitions as specified by the IPPM as far as possible. The IPPM provides a framework [50] and a set of performance relevant metrics which are based on the work of Paxson. Thus, the next Sections will cover the definition of metrics used in this work.

2.2.2 One-way Delay

The *Type-P-One-way-Delay* metric is defined by IETF RFC2679 [1]. The parameters for this metric are *Src* (the IP address of the source), *Dst* (the IP address of the destination), time *T* and *Type-P* denotes the type of IP packets used in the measurement.

The *One-way Delay* (OWD) is the (wire-)time *T* taken from sending the first bit of a *Type-P* packet at *Src* to receiving the last bit of that packet at *Dst*. It is undefined if the packet is not received at *Dst*. Time *T* is therefore a positive real number in seconds or undefined.

2.2.3 Round-trip Delay

The *Type-P-Round-trip-Delay* metric is defined by IETF RFC2681 [3]. The parameters for this metric are *Src* (the IP address of the source), *Dst* (the IP address of the destination), time *T* and *Type-P* denotes the type of IP packets used in the measurement.

The *Round-trip Delay* (RTD), often also called *Round-trip Time* (RTT), is the (wire-)time T taken from sending the first bit of a *Type-P* packet from *Src* to *Dst*, immediately sending back a *Type-P* packet from *Dst* to *Src* and finally to receiving the last bit of that packet at *Src*. It is undefined if the packet is not received at *Dst*, *Dst* did not send back a *Type-P* packet or the returning packet is not received at *Src*. Time T is therefore a positive real number in seconds or undefined.

2.2.4 Jitter

The term Jitter generally denotes the temporal deviation of an event from a predefined point in time. But it is often used in a lax and unprecise manner. Within this work we avoid the use of the term Jitter and rather define correspondent metrics depending on the particular context.

Jitter in the context of Packet Delay measurements. – In this case we use the term *IP Packet Delay Variation* (IPDV) as specified by the IPPM. The IPDV metric is defined by IETF RFC3393 [11]. It can be given for packets within a stream of multiple packets. Then the IPDV is the difference between the OWD of a specific pair of packets out of the packet stream taken at two measurement points. To be more precise: the *Type-P-One-way-IPDV* has the parameters *Src* (the IP address of the source), *Dst* (the IP address of the destination), time T_1 and time T_2 and *Type-P* denotes the type of IP packets used in the measurement.

The IPDV is the difference between the value of the OWD of a packet of *Type-P* sent at T_2 from *Src* to *Dst* and the value of the OWD of a packet of *Type-P* sent at T_1 from *Src* to *Dst* for this two packets being a specific pair of packets out of a stream of packets from *Src* to *Dst*. It is undefined if *Dst* did not receive either one or both packets. The IPDV is therefore a (positive, zero or negative) real number in seconds or undefined. For a clearer understanding on calculating the IPDV, an illustrative definition is provided in Fig. 2.

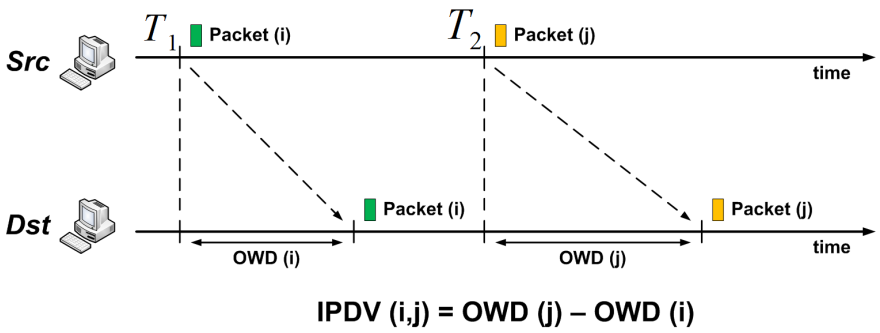


Figure 2: IPDV Metric Definition.

Jitter in the context of the evaluation of the quality of a sender. – In this case we use the term *Tx-Host-Noise*. It can be given for packets within a stream of packets if the desired packet sending times are explicitly known before the packets have been sent. If a packet source should send two packets within an arbitrary, but well defined interval in time, and it actually sends these packets in a time interval that differs from the predefined one, the difference between the desired and the obtained time interval is defined as Tx-Host-Noise. This metric is not provided by the IPPM, but we can define it similar to the IPDV with some modifications. The *Type-P-Tx-Host-Noise* has the parameters *Src*, the predefined packet stream, *Snd*, the sender of the packet stream with IP address, *Dst*, the receiver of the packet stream with IP address, times T_1, T_2, T_3, T_4 and *Type-P* denotes the type of IP packets used in the measurement.

The *Tx-Host-Noise* is the difference between the value of the time interval of two packets of *Type-P* from *Src* that should be sent at T_1 and T_2 by *Snd* to *Dst* and that were sent at T_3 and T_4 by *Snd* to *Dst*. It is undefined if *Snd* did not send either one or both packets or *Dst* did not receive at least one packet. The Tx-Host-Noise is therefore a (positive, zero or negative) real number in seconds or undefined. Fig. 3 depicts the calculation of the *Tx-Host-Noise* metric and its relation to the previously defined *IPDV* metric.

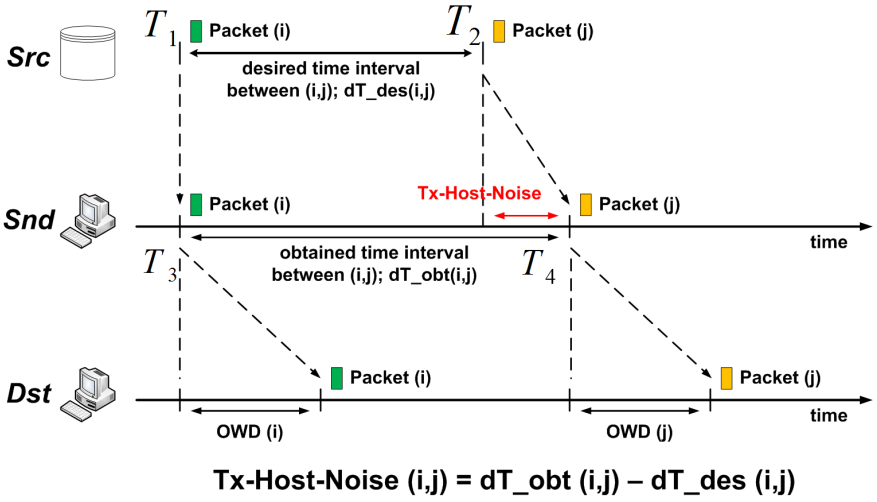


Figure 3: Tx-Host-Noise Metric Definition.

2.2.5 One-way Packet Loss

The *Type-P-One-way-Packet-Loss* metric is defined by IETF RFC2680 [2]. The parameters for this metric are *Src* (the IP address of the source), *Dst* (the IP address of the destination), time *T* and *Type-P* denotes the type of IP packets used in the measurement.

The One-way Packet Loss has the value 0 if the first bit of a *Type-P* packet is sent from *Src* to *Dst* and *Dst* received the last bit of that packet at (wire-)time *T*. The One-way Packet Loss has the value 1 if *Dst* did not receive the complete packet. One-way Packet Loss is therefore either a zero indicating a successful transmission of a *Type-P* packet or a one indicating the loss of a *Type-P* packet.

2.2.6 Hop Count

The Hop Count is a metric strongly related to the Time-To-Live (TTL) field contained in the Internet Protocol (IP) header [24]. While in the original design of the protocol the TTL was meant as a time limiting field to prevent IP packets from looping infinitely inside a network, in practice routers treat the TTL as a hop limiting field. This variation from the original definition was enforced by the performance gain in only decrementing a hop count instead of propagation times [47]. We define the Hop Count as follows:

The Hop Count is the number of signal regenerating devices (such as repeaters, bridges, routers, and gateways) through which IP packets must pass to reach their destination.

2.3 Probe Sampling

As we introduced a couple of singleton metrics in the last Section that reflect the current state of a system being measured we are also interested in variations or relations present within that particular metric. This process of continuously monitoring a specific metric is commonly covered by the item ”Sample Collection”. Especially in the case of *Active Measurements*, at which the system under observation is potentially influenced by the probe traffic, a proper design of the sample collection process is essential (see also Section 2.1). Due to limitations within PlanetLab, i.e. restricted access to specific service routines and limited amount of system resources, that complicate the applicability of *Passive Measurements*, we will focus on *Active Measurements* in the following. Hence, in the remainder of this Section we will cover some design techniques for probe traffic in Active Network Measurements.

The first step in probe traffic design decisions is to consider what behaviour of the network we want to study. A possible intention is to observe a protocol specific behaviour within a network like TCP congestion window sizes or BGP route convergence times. A fairly different intention is to study the properties of a network itself, e.g. the One-way Delay of a network path. This procedure requires a

”Lack of Anticipation” [72], i.e. in a wider sense that a previous sample does not influence a subsequent sample. The probing samples are meant to be unbiased and not skewed from the collection process or measurement respectively. In [50] two general approaches are mentioned for collecting measurement samples:

Periodic Sampling – is the collection of samples separated by a fixed amount of time. This method is quite simple in its handling on the one hand, on the other hand it might lead to undetected changes of network behaviour if the metric being measured exhibits periodic behaviour itself or the injected traffic can drive network components into a state of synchronization. The probe traffic is predictable and therefore suspectible to manipulation [50].

Random Additive Sampling – is the collection of samples separated by randomly chosen time intervals following a statistical distribution. This method overcomes many drawbacks of *Periodic Sampling* concerning bias and synchronization effects but complicates frequency-domain analyses and still remains predictable unless the statistical distribution is exponential [50].

So the recommended practice in [50] is to use *Poisson Sampling*, i.e. probe samples are separated in time following a Poisson (exponential) distribution. This method further permits to compare time averages of the measured metric if the underlying system represents an ergodic stochastic process. Besides that it also offers some other convenient properties (see e.g. [46]). Regardless the apparent advantages of the *Poisson Sampling* or ”Poisson Arrivals See Time Averages” *(PASTA)* [72] approach it cannot be implied to provide optimal results in every single case. In [5] it was turned out that in active probing intrusiveness should be well controlled and kept as low as possible. Therefrom it was recommended to use carefully designed probe patterns dependent on the measurement aim instead of Poisson distributed samples, which cannot be adjusted to form an optimal probe pattern for all kinds of cross traffic on a link. Furthermore, it was shown that *PASTA* only applies to a stream of Poisson packets and cannot be used to justify any inference based on temporal behaviour between probes of packet pairs or trains, where interactions are not memoryless. More on that topic can also be found in [64].

2.4 Bottleneck and Available Bandwidth

In the last Section we figured out the impact of probe traffic modeling. But how can we get an idea of the impact of the network structure and conditions on our measurement results? A fundamental property of a network is the bandwidth it can provide, i.e. how fast a given amount of data can be transferred from a source to a destination via that network. In this Section we deal with the introduction of the terms *Bottleneck Bandwidth* and *Available Bandwith* as well as some common measurement techniques to estimate them.

2.4.1 Synopsis

The maximum bandwidth or capacity B_i of a single link i in a multi-hop, tandem queueing network is the (minimum) time interval T_i in which a packet of size b can be transferred via that link, i.e.:

$$B_i = \frac{b}{T_i}. \tag{1}$$

B_i is referred to as the maximum bandwidth the link can provide. The instantaneous utilization $u_i(t)$ of link i at time $t \in T_i$, i.e. the amount of used capacity, is defined to have the value 1 during the transmission of a single packet and the value 0 if no packet is transmitted. Then the average utilization $u_i(t, t+\tau)$ of link i during the time interval $[t, t+\tau]$ depends on the number of transmitted packets via that link and is defined by

$$u_i(t, t+\tau) = \frac{1}{\tau} \int_t^{t+\tau} u_i(t)dt. \tag{2}$$

The *Available Bandwidth* $A_i(t, t+\tau)$ on link i further denotes the unused capacity during the time interval $(t, t+\tau)$:

$$A_i(t, t+\tau) = B_i[1 - u_i(t, t+\tau)]. \tag{3}$$

Normally a network path consists of multiple network links (hops) $i = 1...N$ and the transmission speed of a packet can only be as fast as the link with the least capacity permits. The capacity $B(t, t+\tau)$ of that so called *Tight link* during the time interval $(t, t+\tau)$ is also denoted as *Bottleneck Bandwidth*. The *Bottleneck Bandwidth* is an upper bound on how fast a connection can possibly transmit data [49] and it is given as

$$B(t, t+\tau) = min_{i=1...N}\{A_i(t, t+\tau)\}. \tag{4}$$

The term *End-to-End Available Bandwidth* is used slightly different from (3). It rather denotes how fast the connection should transmit to preserve network stability, i.e. *End-to-End Available Bandwidth* never exceeds *Bottleneck Bandwidth* and can in fact be much smaller [49]. This a bit confusing distinction must be made if we consider the estimation of Available Bandwidth to be an Active Network Measurement with an end-to-end connection from one host to another. In this case we can only measure the queuing delays (which reflect the Available Bandwidth) undergone by our own probe traffic. However, more likely there will be also competing traffic on each link on the path between our hosts. Hence, what we estimate is not only the unutilized bandwidth on the bottleneck but additionally the queuing delays of all links $i = 1...N$ on the network path. Moreover, it cannot be ensured that during a time interval $[t, t+\tau]$ our probe packets undergo the smallest possible queuing delay at the bottleneck which leads to *End-to-End Available Bandwidth*

being smaller than *Bottleneck Bandwidth*. A more detailed discussion on that topic can be found in [49].

An ongoing effort is spent on development and enhancement of Available Bandwidth estimation techniques today. Many of them are based on dispersion of packet pairs or trains. These techniques are suitable to estimate end-to-end path capacities. A different technique suitable also for estimating the capacity of each link on a path is packet tailgating. In the remainder of this Section we will only show the fundamental working mechanisms of these techniques. A broader overview and comparisons of existing measurement tools and techniques can be found e.g. in [29], [30] or [21].

2.4.2 Packet Pairs and Trains

Packet Pairs bandwidth estimation is based on the idea that two packets transmitted by the sender are passing router queues along their path to destination in the same order as they were sent. That idea requires the router queues to work in FIFO or store-and-forward mode, which is usually the case in the Internet. The tight link on the path, i.e. the link with less Available Bandwidth has the capacity ρ_B. Thus, the term

$$Q_b = \frac{b}{\rho_B} \tag{5}$$

is the amount of time required to forward a packet of size b through the tight link. If the two probe packets are sent with a time spacing $\Delta T_s < Q_b$ between them and they pass the bottleneck, their spacing will be spread out in time. At the destination the delay variation between both packets is measured and compared to the original one at the sender. The amount of degradation of the delay variation at the destination then allows to estimate the Available Bandwidth. Fig. 4 illustrates this mechanism. The estimate of Available Bandwidth from delay variations can be done with various techniques, which ones to describe is beyond the scope of this report. Additional information regarding *Packet Pairs* technique can be found e.g. in [9], [31] or [7].

Since *Packet Pairs* do not capture the temporal queuing behaviour a method called *Packet Trains* was introduced to overcome this shortness. *Packet Trains* do not estimate Available Bandwidth only from a pair of packets but rather calculate one estimate from multiple pairs of packets (a train). The time spacings of packets in successive trains are altered in order to get a convergence to Available Bandwidth or an average of it from receiver packet-spacings. The main difference between *Packet Pairs* and *Packet Trains* is that *Packet Trains* dispersion values potentially carry information of cross-traffic variance which allows for better detection and characterization of that cross-traffic. *Packet Trains* techniques are used e.g. in [38] or [28].

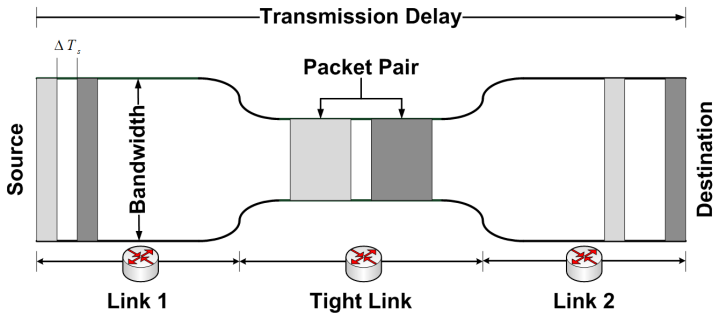


Figure 4: Packet Pairs Method.

2.4.3 Packet Tailgating

While the previously presented Available Bandwidth estimation techniques are aimed to measure end-to-end queuing behaviour on network paths, *Packet Tailgating* is another method that is further capable to detect per-hop link capacities and Tight link locations. *Packet Tailgating* uses Packet Trains consisting of large packets whereby each large packet is immediately followed by a small packet. The large packets are sent with a limited TTL and exit somewhere on the path at the target node. The smaller packets are travelling forward to destination while providing timing information of the delay dispersions. This technique can also be used to circumvent problems that arise with multi-channel bottleneck links as described in [49]. Multi-channel links occur on network paths with separate physical channels or routers that balance load across multiple links, yielding to misleading estimates with Packet Pairs. The mechanism of *Packet Tailgating* is illustrated in Fig. 5. Recent work based on *Packet Tailgating* can be found e.g. in [33], [18] or [53].

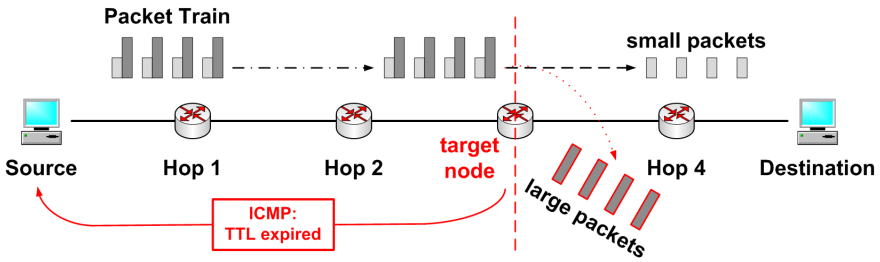


Figure 5: Packet Tailgating Method.

2.5 Clocks and Timers

Many metrics and measurement techniques presented in the previous Sections rely on accurate time information. So, for sound Internet measurements it is important to understand the different types of uncertainties and errors introduced by imperfect clocks. In this Section we introduce terminology related to clocks and describe problems that arise in connection with IP measurements. We further give an overview of common clock sources and methods for synchronizing network hosts while we end this Section looking on techniques for handling screwed-up measurement traces.

2.5.1 Clock Terminology

One of the first protocols used in the Internet is the Network Time Protocol (NTP) [39]. It defines a basic nomenclature regarding clock characterization which was adopted in the IPPM framework [50]. In the remainder of this Section we will pickup the most important definitions in a brief manner to provide a clear terminology.

Offset – denotes the difference between the time reported by a clock and the ”true” time at a particular moment. True time in the sense of the IPPM is the *Universal Time Coordinated* (UTC).

Accuracy – is the absolute deviation of a clock’s Offset to zero at a particular moment, i.e. a clock is ”accurate” if its Offset is exactly zero.

Skew – is the first derivative of a clock’s Offset with respect to true time. More descriptive, Skew is the frequency difference between the clock and true time.

Drift – is the second derivative of a clock’s Offset with respect to true time. This implies that the clock’s Skew can vary.

Resolution – of a clock is the smallest unit by which the clock’s time is updated. Resolution is relative to the clock’s reported time and not to true time, i.e. that the clock updates its notion of time in arbitrary time increments which are not necessarily the true amounts of time between updates.

All of the above definitions can also be used when comparing clocks, i.e. their values are not referenced to UTC but rather to another clock’s reported time. The IPPM suggests the use of the term ”relative” in this case. Hence, e.g. the relative Offset between two clocks then is the difference in time reported by the two clocks. The only exception is the case of comparing clock Resolutions. The wording ”joint” Resolution is used instead of ”relative” Resolution, denoting the sum of the Resolutions of both clocks. Another definition is needed when comparing the Accuracy of two clocks:

Synchronization – between clocks is present if their relative Offset is zero. Note that two clocks can be highly synchronized but they can still be far away from true time (UTC).

2.5.2 Time Sources

Now having defined an appropriate wording related to clocks we will take a closer look on how time is provided on personal computers (PCs), which are probably the most frequently used platforms for Internet measurements. In this Section we will give an overview of commonly available time sources and their characteristics on x86/x86_64 architectures.

The basic time (or tick, respectively) sources on PC hardware are generally oscillator circuits driven by quartz crystals. The crystal provides a pulse at a given frequency which is fed into different channels, each having a counter register. The counters are incremented at multiples of the crystal pulses allowing to provide different clock signals to several components.

The x86/x86_64 architecture has evolved a long way since its inception and different time sources were introduced until today, each of them having its advantages and drawbacks. Table 1 gives an overview of tick sources and their Resolutions available on contemporary hardware.

Time Source	Resolution
Programmable Interrupt Timer (PIT)	1,000.15 Hz
Real Time Clock (RTC)	8,192 Hz
Power Management Timer (PMT)	3.58 MHz
High Precision Event Timer (HPET)	10 MHz
Time Stamp Counter (TSC)	up to 4 GHz (Processor speed)
Local Advanced Programmable Interrupt Controller (LAPIC)	up to 400 MHz (FSB speed)

Table 1: x86 Time Sources.

Using the tick source with the highest Resolution is not implicitly the solution that provides the best results. Many other factors influence the stability and reliability of a time source. E.g. environment temperatures, processor power states or interrupt latencies can cause a clock to drift over the time (see e.g. [8] or [37]. However, if an HPET device is present, it should be the preferred choice as time source [8].

Fairly different approaches of timekeeping associated with its own problems appear when performing time-related measurements within Virtual Machines (VMs). Due to the variety of deployed virtualization solutions it is difficult to give an ab-

solute overview of time source handling within VMs. But it should be stressed to handle such measurement results with care and to investigate the underlying time synchronization mechanisms in the forefield. A more detailed description regarding timekeeping issues within a commercial virtualization solution can be found in [23].

2.5.3 Software Timestamping

As mentioned in the previous Section the choice of a viable tick source is not the only issue affecting the ”quality” of time stamps in a measurement. An essential instance lying on top of the tick source is the kernel of the used operating system. The kernel is responsible for handling and coordinating all system resources, i.e. the kernel finally decides e.g. when to assign a time stamp to an incoming IP packet. In this section we will cover several aspects of the Linux kernel and its timer system.

Linux is a multitasking and multiuser operating system. The management of processes and threads is accomplished by the process scheduler, i.e. it determines when physical processors should be assigned and to which processes. The granularity of the scheduler is at that lower bounded by the underlying kernel timer system.

The legacy Linux timer system is the *Cascading Timer Wheel* (CTW). It is based on periodic interrupts (jiffies) from which differences in time are calculated by counting the jiffies. The maximum update interval of the CTW is defined by the ”HZ” value compiled into the kernel. Since the introduction of the 2.6 kernel the maximum resolution can be set to 1000 Hz, i.e. processes can be scheduled with a granularity of 1 ms.

In 2006 the *high resolution timers* (*hrtimers*) subsystem was integrated into the Linux kernel [14]. It comes with a gigahertz resolution which theoretically enables the kernel to schedule processes with a granulartiy of 1 ns. In practice there is no common tick source available providing a stable resolution in the nanosecond regime. Nonetheless the *hrtimers* time system gains a significant improvement over the legacy CTW system. When performing Internet measurements on Linux platforms it should be ensured that *hrtimers* support is enabled in the kernel.

Another aspect of process scheduling that inherently comes along with timing precision is the scheduling algorithm. A scheduling algorithm is called nonpreemptive if, once a process has been given the processor resources, the process runs until it has finished. The algorithm is called preemptive if that process can be temporarily suspended from its execution. The latter is also known as real-time preemption. More generally, real-time enabled systems allow a more or less deterministic scheduling, meaning that the response times of processes can be guaranteed. A fairly sophisticated real-time solution for Linux is the *RT-Preempt Patch* [58] empowering the kernel to work in complete preemption mode. Real-time preemption helps to improve timing precision in Internet measurements since it reduces

the latency of `gettimeofday()` system calls [58], which are normally used to generate timestamps. This bearing directly governs the *Tx-Host-Noise* metric we already covered in Section 2.2.4.

2.5.4 Clock Synchronization

In addition to the previously mentioned tick source precisions and kernel timer resolutions some Internet measurements require highly synchronized measurement endpoints. With the gaining attraction of multimedia services in the past years stringent delay requirements became an important issue. Packet based telephony for example does not only suffer from Jitter but also impairs from high end-to-end delays. Especially One-way Delay is a critical design parameter which has to be gauged in a VoIP environment. Due to its prevailing dependence on Synchronization One-way Delay measurements are still a challenging task. In this Section we will observe two time synchronization techniques commonly used today.

Since the early beginnings of the Internet time synchronization has always been an important issue. So the IETF already defined the Network Time Protocol in 1985 which was the successor of the *DCNET Internet Clock Service* specified in 1981. NTP is designed to let clients contact a time server which should accurately provide the current time. The Accuracy of the time servers themselves is stacked in a self-organizing hierarchical structure. *Stratum 0* or reference devices are assumed to be accurate and directly feed *Stratum 1* or primary reference servers, which they are attached to. Reference devices are e.g. atomic clocks or GPS receivers. Each server synchronizing to the *Stratum 1* server over a network path then is called a *Stratum 2* or secondary reference server. The Stratum level is consecutively increased with every additional intermediate node in the chain. An illustration of the hierarchical layer structure is shown in Fig. 6. Clients synchronizing to an NTP server then run a daemon that perpetually sends timestamped requests to the time server which likewise returns a timestamped response piggybacked by the request's time of receipt. If the propagation delays from client to server and vice versa are nearly the same, the client can determine the relative Offset to the reference time of the server. To adjust a possible clock Offset the client can either directly set its current time to the estimated reference time or it varies its clock frequency to achieve convergence to the reference time in the future. The latter means in fact changing the client's clock Skew.

This behaviour already shows the suboptimality of NTP for One-way Delay measurements. NTP is primarily aimed for long-term time synchronization in ranges of minutes or hours whereas e.g. One-way Delays often have a dynamic range in the order of milliseconds or below. NTP-induced changes of the local clocks's Offset could lead to heavy discontinuities, clock frequency adjustments can produce considerable inaccuracies and the particular network delay dependency makes NTP impractical for time calibration in many IP performance measurements (also refer to [50]).

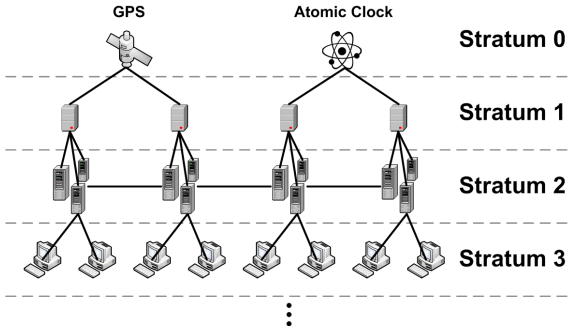


Figure 6: NTP Hierarchy.

A more sound approach for synchronizing measurement hosts is the use of the Global Positioning System (GPS). GPS includes at least 24 satellites that orbit the earth at a height of 20,200 km. The satellites broadcast two spread spectrum PN-coded waveforms on two carrier frequencies (L_1=1,575.42 MHz and L_2=1,227.6 MHz). Only the signal on L_1 with a chip-rate of 1,023 Mcps is for public use, the other one is reserved for the US Department of Defense (USDOD). Each satellite is equipped with rubidium or cesium oscillators referenced to UTC maintained by the United States Naval Observatory (USNO). GPS receivers can track up to 12 satellites and mostly require Line-of-Sight propagation, i.e. they must be placed outdoors with direct view into the sky. Receivers suitable for time synchronization provide a 1 pulse per second (PPS) signal which is derived from an average of all tracked satellites. The PPS signal is turned into a time-of-day format and can be directly fed in a computer via the RS-232 or USB interface. On May 2, 2000 USDOD turned off the interference signal (Selective Availability Program (SA)) which now allows to receive a PPS signal output with a standard deviation of 10 ns or less [35]. Fig. 7 shows the PPS signal phase plot from a typical GPS timing receiver to UTC (NIST) before and after the SA deactivation (the signal is plotted against the *Modified Julian Date* (MJD)).

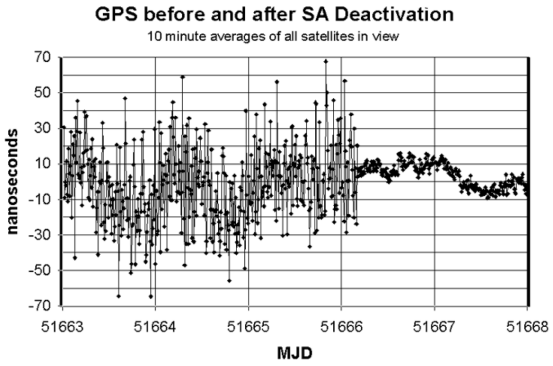


Figure 7: GPS PPS Signal (from [35]).

2.5.5 Dealing with Clock Imperfections

In the previous Sections we showed how to improve clock Accuracy and Synchronization of measurement systems. Unfortunately it is not always possible to achieve optimal conditions. Heavy manipulation of operating system routines may be restricted due to user permissions or the lack of physical access to measurement hosts. While we anticipate that obtaining accurate relative Offsets in One-way Delay measurements is only possible with proper Synchronization or instead by using Round-trip captures, clock Skew can be assessed in a post processing routine. In the remainder of this Section we therefore present techniques and algorithms aimed on detecting clock adjustments as well as clock Skew removal from measurement traces.

In [48] Paxson developed an algorithm for detecting clock adjustments. It is based on bidirectional One-way Delay measurements that exhibit equal but opposite level shifts. Paxson also concedes various problems with that algorithm resulting in false positives or negatives. Since our work is targeted on unidirectional measurements we do not further adopt Paxsons algorithm and settle for trivial clock adjustment detection. Trivial clock adjustment detection is possible if the adjustment is relatively large and occuring in an abrupt manner. An instantaneous and permanent shift in delay within a packet stream is likely to indicate an adjustment of a host's clock. To distinguish between a clock adjustment and a route change, which also can lead to abrupt time shifts, it is desirable to investigate the number and labels of the traversed hops as well. If the route remains constant during the time shift there is a high possibility that the shift is caused by a clock adjustment. A further trivial scenario is a backward clock adjustment. If a time stamp occurs that already has been captured before, time causality is violated most probably induced by a backward clock correction. As mentioned before in Section 2.5.4 time adjustment techniques are often based on clock frequency variation which slowly shifts the clock's Offset towards the target time. While those procedures in fact change the clock's Skew a more important challenge is to assess this type of uncertainties. Also note that Skew might not only appear in consequence of clock adjustment algorithms but is rather a "feature" of many computer clocks due to the assembly of cheap components.

Clock Skew might not seem to be a problem in various measurement scenarios due to its relatively small values. But if we assume variations in One-way Delay to reflect the congestion levels within a network the induced error may lead to large misinterpretations. Paxson suggests in [48] a Skew removal algorithm that is based on a probabilistic minima test and robust line fitting techniques. He first partitions a One-way Delay trace of N samples into $\sqrt{N}$ segments and picks the minimum delay of each segment. This procedure is used as a kind of noise removal. The next step is to detect if a decreasing trend in One-way Delays is present. If the median of the slopes of all possible pairs of "de-noised" OWDs is negative, a decreasing trend

is detected. Then a cumulative minima test (see [48]) is applied to test whether the previously detected trend is probabilistically likely or not. If the test is passed the median of the pairwise slopes of all possible "de-noised" OWDs is assumed to be the relative Skew.

Moon et al. presented in [40] a method called Linear Programming Algorithm (LPA) to remove relative Skew from OWD traces. It assumes a clock as a piecewise continuous function that is twice differentiable. The basic idea is to fit a line representing the Skew, that lies under all OWDs, but as closely to them as possible. The objective of the LPA is to find

$$\begin{aligned} &min\{\sum_{i=1}^{N}\Big(d_i-(\alpha-1)t_i^s+\beta\Big)\} \quad s.t. \\ &d_i-(\alpha-1)t_i^s+\beta\geq 0, \quad 1\leq i\leq N \end{aligned} \tag{6}$$

where d_i denotes the estimated delay after Skew removal, t_i^s is the time duration between the first and the i-th packet with respect to the sender clock C_s, the Skew of C_s is $\alpha = \frac{d}{dt}C_s(t)$ and β defines the time duration between the first packet measured at C_s and the receiver clock C_r. Furthermore C_r is assumed to be the "true clock" meaning $\frac{d}{dt}C_r(t) = 1$. Simulations applying synthetic delays in [40] compared LPA with Paxson's algorithm and showed a better performance of LPA in terms of bias and variance of the estimated clock Skew.

While the aforementioned algorithms assume constant clock skews without adjustments and drifts during measurements, Wang et al. extended Moon's algorithm to be applicable in this case [71]. They introduce the "K-segmentation Optimized Top-Down Algorithm" (K-OTDA) to detect clock adjustments that cause time uncontinuity. The K-OTDA algorithm segments an OWD trace into several parts which are then again assumed to be piecewise continuous. Hence, each segment's clock Skew can be separately estimated using (6). Despite a sophisticated segmentation procedure finding the true number of segments is stated yet to be an open issue in [71]. Also computational complexity increases from $O(N)$ for Paxson's and Moon's algorithm to $O(N^2)$ for Wang's algorithm.

Although many other Skew estimation techniques had been proposed, the key problem still remains: to differentiate between clock imperfections and network dynamics. We emphasize to stringently rely on proper Synchronization and clock Accuracy rather than on subsequent data processing.

2.6 Topology Discovery

Besides attaining precise knowledge of delays within a network, gathering topology information is an equally important issue in many cases. Unfolding a network's structure may indeed allow to predict delays with subject to routes between connection endpoints. We guess e.g. that on intercontinental WAN links approximately one third of the OWD is caused by queuing delays in routers (see Section 4.2.2).

With increasing Hop Counts the particular queuing delays will add up to a significant percentage of the overall delay. Route changes due to load-balanced paths may also cause considerably different amounts of delay. Getting knowledge of such network characteristics benefits many network design decisions or traffic engineering tasks. In this Section we will treat aspects of Internet topology discovery and pitfalls coming along with that issue. As we are focused on Hop Counts in this work we will address this subtopic in more detail.

The basic tool used in Internet topology discovery is the well-known traceroute (TR) [27]. It relies on the Internet Control Message Protocol (ICMP) as defined in RFC792 [51]. TR uses TTL-limited probe packets that cause ICMP time-exceeded responses from each router on the path that receives an IP packet with TTL exhibiting at least a value of 1. The TTL is steadily increased until the probe packets reach their destined receiver. Thus, every router on the path from sender to receiver can be traced. Sadly, in practice this approach does not always behave well. Early measurements in [15] exploiting the aforementioned mechanism reported that ISPs sometimes configure their routers to not decrement TTLs across their infrastructures. Also some misconfigured routers were recognized inferred from ICMP responses of multicast and private addresses. Another hardly to assess problem was the alias resolution, i.e. the recognition of interfaces belonging to one and the same router. Moreover, in [36] routers with multiple aliases were reported to cause traceroute to abort after 30 hops (the default maximum) on 15% of the measured paths. Furthermore, about 50% of the explored path endpoints did not even respond with ICMP messages. A reason for that finding are firewalls that block traceroute packets or the routers do not participate in ICMP at all.

Another, somehow dismissed so far, option of IP, record route (RR), was used in [61] in combination with traceroute like techniques to overcome some of the prementioned misbehaviours. The RR option enabled in the IP header tells a router to insert its own Internet address as known to the environment into the IP options field. A main limitation of this method is that only 9 entries can be held in the header. Routers that are passed after the limit is reached are not recognized anyway. To circumvent this limitation a measurement platform with sufficient network diversity, such as PlanetLab, is required. It was reported in [61] that originating from PlanetLab approximately 87% of IP addresses were reachable in 9 hops. This coverage rate might be much better until now since new PlantLab nodes are frequently installed. Beyond using the IP RR option the measurements were also embedded into Transport Control Protocol (TCP) [25] streams to reduce firewall blockings. Despite also defeating diverse problems as reported in [61] the RR method however allows the detection of so called hidden routers that never decrement TTLs. Such routers are supposed to be part of Multi Protocol Label Switching (MPLS) [55], [56] clouds. MPLS is a protocol that allows to build up predefined connection-oriented paths through a packet network for the purpose of faster routing and traffic engineering. Entering the MPLS cloud at ingress routers, traffic flows are categorized and assigned a label which is recognized and used for

packet forwarding by all other routers within that cloud. IP header TTLs will be only decremented again as the traffic leaves the cloud at an egress router. Instead, MPLS-enabled routers maintain their own TTLs within the MPLS label. Many routers thereby can explicitly be configured to hide the network's topology by disabling TTL decrementation and therefore making the MPLS cloud appearing as a single hop. A further study in [60] using again RR measurements revealed 0.3% hidden routers out of a set of 100,256. Also 8,550 persistent routing loops out of 376,408 source-destination pairs were detected, preventing packets from reaching their destination. A further reason for the classical traceroute to fail identifying correct paths is load-balancing as formerly mentioned in [4]. Load-balancing decisions are often based on the so called *five-tuple* of fields from the IP, TCP or UDP headers: Source Address, Destination Address, Transport Protocol, Source Port, and Destination Port. Common strategies are per-flow, per-packet or per-destination load balancing. Since routers mostly balance equal-cost paths, many nodes are not detected by the classical traceroute as described in more detail in [4]. RR traces on the other hand are assumed to work load-balancing aware [60].

The presented examples demonstrate the severity of Internet topology discovery and the list of uncertainties can be further extended. To obtain a rather accurate view on a network's topology more sophisticated tools, besides traceroute, are needed. Dealing with Hop Counts therefore requires a high degree of carefulness. Using IP RR and TCP embedded probe packets is potentially preferable over the classical traceroute tool. Also other traceroute variants like *Paris traceroute* [4] or *paratrace* [52] may be suitable.

3 Methodology

The variety of metrics and techniques presented in Section 2 enable us to understand the fundamentals of network measurements and especially on what to turn our attention to. Certainly not all conditions will be always ideal and often rely on circumstances we cannot affect in particular. Usually a tradeoff between accuracy, measurement objectives, complexity and the prevailing conditions must be found. In this Section we present our methodology for collecting and evaluating path characteristics in PlanetLab.

3.1 Measurement Apparatus

PlanetLab serves us a pool of world-wide distributed measurement hosts. Since not all nodes are suitable for our purposes we need to obtain coarse topology information and run a candidate pre-selection in the first step. We then add selected nodes to our slice and perform the measurements. For the ease of administration and better scrutiny we perform all actions originated from the *planetlab01.tkn.tu-berlin.de* node deployed in our department. In this part we provide information on how we obtained topology information and collected measurement samples.

3.1.1 Network Exploration

A comprehensive list of PlanetLab nodes including frequently updated node state informations like Boot States, Disk Utilizations or Tx and Rx-Rates, is provided by the CoMon project [43]. CoMon runs node-centric and slice-centric daemons on each node participating in the monitoring system. This daemons collect and report raw data every five minutes to a central database. Metafiles and statistics are then generated and prepared for the use through a CGI query interface which also provides several formatting expressions for user-provided output and selection criteria.

We use CoMon to obtain a list of all nodes that are alive and do not suffer any problems, turning our attention especially on low clock Skew values. Next we pass the mentioned list to a (classical) traceroute routine which records the paths and RTTs to all obtained nodes. The traceroute routine is further coupled with an interface to *hostip.info* [20], a community-based geolocation project which provides a database API to geographic coordinates and locations of IP addresses. For the portion of resolvable PlanetLab nodes within the *hostip.info* database we store the retrieved coordinates and calculate the airline distance d from *planetlab01.tkn.tu-berlin.de* to the corresponding node's location using the haversine formula [62]:

$$d = R \cdot 2 \arcsin \left\{ min \left[1, \sqrt{\sin^2 \left(\frac{\Delta\phi}{2} \right) + \cos \phi_A \cos \phi_B \sin^2 \left(\frac{\Delta\lambda}{2} \right)} \right] \right\} \quad (7)$$

where d denotes the spherical distance between points A and B; ϕ_A, λ_A and ϕ_B, λ_B are the latitude and longitude of A and B respectively, $\Delta\phi$ is the latitude difference as well as $\Delta\lambda$ is the longitude difference and $R = 6,371.0087714\ km$ is the mean radius of the earth as defined by the *Geodetic Reference System 1980* (GRS80) [41]. We then use (8) to estimate the related fiber distance l between A and B [26].

$$l = \begin{cases} 1.5d, & d < 1000\ km \\ 1500\ km, & 1000\ km < d < 1200\ km \\ 1.25d, & d > 1200\ km \end{cases} \tag{8}$$

Further on, territorial information is also obtained where possible.

Unfortunately, conducting some random spot tests, we unveiled several inaccuracies in the *hostip.info* database. E.g. for router *oak-hpr–svl-hpr-10ge.cenic.net* (*137.164.25.9*), lying on the path between *planetlab01.tkn.tu-berlin.de* (Berlin, Germany) to *planetlab7.millennium.merkeley.edu* (Berkeley, CA, USA), *hostip.info* returned the right localization, that is [*UNITED STATES (US), Cypress, CA*], but the associated coordinates were provided by $(Lat, Lon) = (40.7488°, -73.9846°)$, which in fact correspond to a location in New York City (NY, USA). This result was obtained on January 6th 2009. Later on (May 2009), we found the same database entry being corrected in the meantime, i.e. the returned localization [*UNITED STATES (US), Cypress, CA*] matched the corresponding coordinates, which are correctly $(Lat, Lon) = (33.8163°, -118.038°)$. Nevertheless, the given example demonstrates that several obtained node coordinates may remain somewhat unreliable.

However, using a more sophisticated topology discovery mechanism (see Section 2.6) and a reliable geolocation database remains future work, though. The (delay) constraint based geolocation technique presented in [16] appears to be a promising approach to align *hostip.info* data in order to verify its integrity.

3.1.2 Data Aquisition

In order to measure the path characteristics and collecting required data towards a selected PlanetLab node, we use a client-server based architecture. A TCP control connection searches for unutilized ports and steers the probe engine and the capturing instance. Our probe engine is completely based on the Packet Capture (PCAP) [65] file format which allows for convenient storage, processing and analysis of the measurement data. The packet generator calculates Poisson distributed timestamps according to [50] and generates as small as possible 40 Byte UDP/IP probe packets (20 Byte IP-header + 8 Byte UDP-header + 12 Byte payload), each containing a 64 Bit timestamp and a unique 32 Bit sequence number in the UDP payload. Timestamps and sequence numbers thereof are needed for exact identification of each probe packet in later processing. Randomness of the Poisson distribution thereby can be either set constant (predefined seed) or variable (automatic

seed). Timestamps and sequence numbers then will further be used for encapsulation of each probe packet into a PCAP Record Header and finally the PCAP Global Header is generated. Thereafter all data is concatenated in chronological order and the probe stream is stored to a file. For performing the measurement we load the previously (or an arbitrary) generated file into a PCAP player routine which recalculates the required time intervals for sending. The player detects and creates all required sockets whereupon it starts sending the packets using either `usleep()` or `nanosleep()` to adhere the correct timing. If supported, the usage of raw sockets with `IP_HDRINCL` can be forced for slightly better performance. This option prevents modification and recalculation of the IP header through the network stack. To avoid potential problems with NTP time shifts the usage of `CLOCK_MONOTONIC` can be forced which guarantees continuous clock ticks. Right before the player starts sending, the TCP control part triggers a tcpdump capture on the sender as well as on the receiver side. After the measurement has finished both tcpdump traces are stored to PCAP files as well, thus providing information on effective transmission and receive times of probe packets. Fig. 8 illustrates again the entire measurement framework architecture.

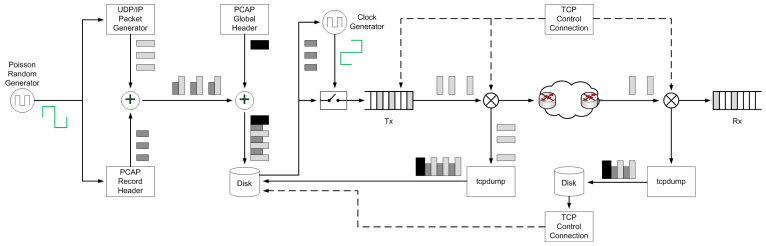


Figure 8: Measurement Framework Architecture.

3.1.3 PlanetLab Issues

Although PlanetLab offers researchers a great networking platform some limitations must be accepted. An essential point to mention is that PlanetLab nodes are operated within virtual machnines. Each node acts as guest operating system controlled and scheduled by a hypervisor which is the lowest and most privileged layer. Normally the hypervisor is administrated by the PlanetLab support and users inside the guest system have no access to that layer. Presently we found Fedora *vserver* virtualization environments to be in use on all tested nodes but *XEN* is also considered in the PlanetLab documentation. Due to the prementioned architecture and the lack of sufficient permissions we were not able to use other Linux kernels than the existing one. This may complicates optimization processes or precludes special requirements in some cases. However, we were comfortable with the finding that *hrtimers* support comes built-in. *Safe raw sockets* are another limitation to handle. An application using raw sockets, such as ping or traceroute, requires

explicit bindings to local ports for proper function (see e.g. [6]). This restriction hinders the utilization of some existing applications. For example we were not able to run *tcpreplay*, a PCAP based packet replay toolsuite [13], which also relies on raw sockets. Last but not least there are also restrictions regarding resource sharings such as CPU utilization, memory consumption, traffic volumes and bandwidth which possibly should be considered.

3.1.4 Extended Studies

In particular cases we extended our measurements to investigate real-world application performance. For this purpose we adapted our framework (see Section 3.1.2) and integrated a VoIP packet generator. The VoIP flow is unidirectional only, which is quite sufficient for our intentions. As a baseline scenario we utilize a classical *G.711* [67] coded voice stream carried over *Realtime Transport Protocol* (RTP) [59] and *Session Initiation Protocol/Session Description Protocol* (SIP/SDP) [57], [17] signalling. SIP/SDP thereby is not fully functional but integrating those message flows allows for convenient detection and analysis of VoIP calls in PCAP based network protocol analyzers like *Wireshark* (see e.g. [70]). The packet generator takes an arbitrary G.711 μ-law or a-law coded audio file and splits the voice data into 160 Byte packets which corresponds to the default vocoder packetization interval of 20 ms. Each voice segment is then appended to an RTP header and embedded into UDP/IP resulting in a 200 Byte VoIP packet. Thereafter the call signalling SIP `INVITE` with SDP, SIP `200 OK` with SDP and SIP `ACK` messages are generated and finally all data is stored in a PCAP file for later use. If the VoIP measurement is to be performed the packet player loads the corresponding file, extracts the timing information and sends a 200 Byte RTP/UDP/IP voice packet every 20 ms, thus consuming 80 kbps conversational bandwidth on IP layer. Outgoing and received VoIP packets are likewise captured with tcpdump in order to provide sufficient information for later analysis. Table 2 summarizes the basic VoIP parameters.

Vocoder	**Voice Bandwidth**	**Packetization Interval**
ITU-T G.711	64 kbps	20 ms
Voice Payload	**Voice Transport**	**Transport Packet Size**
160 Byte	RTP/UDP/IP	12+8+20 Byte
Packets Per Second	**Transport Bandwidth**	**Bandwidth Per Conversation**
50	16 kbps	80 kbps

Table 2: VoIP Baseline Parameters.

Further on, we also used *PathChirp* [54] and *STAB* [53] for Available Bandwidth estimation. With that information on network congestion levels we are able to locate possible sources of application performance degradations and can distinguish between host-induced or network-induced points of failure.

3.2 Evaluation Strategies

After collecting measurement samples further processing comprises the extraction of useful information and evaluation of the dataset. As described in the previous part each measurement outputs three PCAP files which we use to calculate several metrics. We also did some validation effort to estimate the accuracy of the probe engine and the underlying operating system on PlanetLab nodes. In this part we describe the extraction of used metrics and the validation process.

3.2.1 Path Characteristics

In the case of one-way path characteristics measurements we investigate a particular path using a Poisson distributed probe stream. The probe stream is generated comprising $i = 1...n$ UDP packets p_i, each tagged with a unique sequence number i and a timestamp. During the measurement the probe engine stores three PCAP files which include a) the desired packet times $t_{i,Des}$, b) the packet times at transmission $t_{i,Tx}$, and c) the received packet times $t_{i,Rx}$. We calculate the following path characteristics from these traces (according to Section 2.2):

UDP-One-way-Delay – OWD_i of packet p_i is calculated as the time the packet travels from sender to receiver including the relative Offset between both nodes, i.e.

$$OWD_i = t_{i,Rx} - t_{i,Tx} \quad 1 \leq i \leq n. \tag{9}$$

UDP-One-way-IP-Packet-Delay-Variation – $IPDV_i$ between packets p_i and p_{i+1} is the difference of inter-packet intervals of p_i and p_{i+1} at transmission and reception. Since measurement hosts are not necessarily synchronized and thereof the measured OWDs might be incorrect, we avoid using OWDs in IPDV calculations and rather use the packet timestamps at transmission and reception which gives the correct results, even in the unsynchronized case, i.e.

$$IPDV_i = \{t_{i+1,Rx} - t_{i,Rx}\} - \{t_{i+1,Tx} - t_{i,Tx}\} \quad 1 \leq i \leq n-1. \tag{10}$$

UDP-One-way-Packet-Loss – OPL_i is counted as zero if packet p_i was captured at the receiver, i.e. $t_{i,Rx}$ exists, or it is counted as one if $t_{i,Rx}$ does not exist, therefore

$$OPL_i = \begin{cases} 0, & t_{i,Rx}\ exists \\ 1, & otherwise \end{cases} \quad 1 \leq i \leq n. \tag{11}$$

In order to investigate the structure of *OPL*, i.e. for instance burstiness and frequency of packet loss, we further adopt the "loss-distance" metric derived from One-way Packet Loss and as specified by IETF RFC3357 [32]. Note, that this metric requires monotonically (by one) increasing sequence numbers of successive probe packets.

UDP-One-Way-Loss-Distance-Stream – $OWLDS_i$ has the value of a "loss-distance" in terms of a difference of the current sequence number i to the sequence number of the previously lost packet, if the current packet p_i is considered lost, i.e. the corresponding OPL_i is equal to one. Otherwise, if packet p_i was received successfully, i.e. OPL_i is equal to zero, $OWLDS_i$ is counted as zero. Hence,

$$OWLDS_i = \begin{cases} i-j, & OPL_i = 1 \\ 0, & OPL_i = 0 \end{cases} \quad 1 \leq i \leq n. \tag{12}$$
$$j = max\left\{k | OPL_k = 1, k = 1...i-1\right\}$$

Hop Count – HC_i of packet p_i between sender and receiver is taken as the decrement of the IP TTL at transmission, $TTL_{i,Tx}$, and the IP TTL at receiption, $TTL_{i,Rx}$, i.e.

$$HC_i = TTL_{i,Rx} - TTL_{i,Tx} \quad 1 \leq i \leq n. \tag{13}$$

3.2.2 Clocking Inaccuracies

In order to estimate clock resolutions, latencies of the process schedulers as well as processing delays we partly reuse our player routine. Through comparing a set of predefined packet timer intervals and the deviations to recorded time-of-day values between these intervals we obtain the "noise" induced by the system. While in fact no packet is sent during this process the measured values can be taken as a lower bound on the noise that occurs at transmission. Therefore we use that results to asses the *TX-Host-Noise*.

Tx-Host-Noise – THN_i between packets p_i and p_{i+1} is defined similar to (10), except that it is the difference of desired inter-packet intervals of p_i and p_{i+1} and the ones at transmission, i.e.

$$THN_j = \{t_{i+1,Tx} - t_{i,Tx}\} - \{t_{i+1,Des} - t_{i,Des}\} \quad 1 \leq i \leq n-1. \tag{14}$$

Naturally, measurement timestamping precision also suffers from noise on the receiver side. However, our approach does not directly provide us information on that quantity. But the aforementioned lower bounds inferred from synthetic measurements can be used as a rough estimation on that value.

3.2.3 Voice over IP Quality

A fairly more complex issue is the assesment of Voice over IP quality. Three primary metrics affecting voice quality are delay, jitter and packet-loss. Besides sophisticated psycho-acoustic model based techniques to measure subjective call qualities, e.g. *Mean Opinion Scores* (MOS) [68], a comparatively simple and rather objective evaluation criterion derived from ITU G.114 [69] exists. It defines three bands of One-way Delays in conjunction with a One-way Packet Loss rate for classifying the quality of voice calls. The maximum allowed delay therein amounts to 400 ms associated with a maximum packet-loss rate of 5%. A detailed definition of the three bands is depicted in Fig. 9. It should be noted that the G.114 recommendation is oriented for national and international scale communication networks. Inside private networks One-way Delays can be slightly higher, in the region of 250 ms, while still maintaining acceptable voice quality.

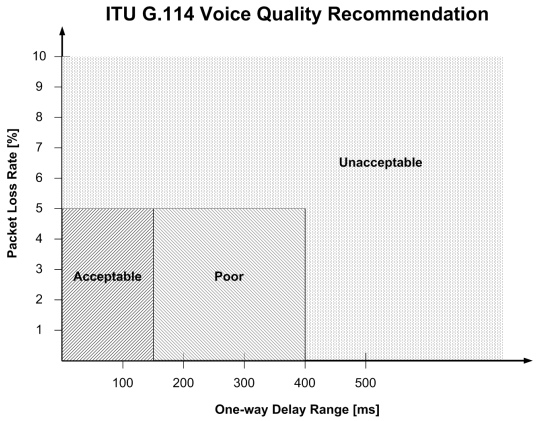


Figure 9: ITU G.114 Voice Quality.

Moreover, the Delay ranges in Fig. 9 must be understood as ear-to-mouth delays. I.e. the overall acceptable One-way Delay is composed of several fixed and variable delay sources in the network. At the sender side it takes a fixed fraction of time for the vocoder for compressing a PCM block and packetization of the compressed speech. For example the default packetization interval of the G.711 vocoder amounts to 20 ms (also see Section 3.1.4). The voice packet then is encapsulated in IP packets and queued for transmission which usually varies in time. At transmission a further fixed serialization delay occurs to clock the voice frame onto the network interface. Upon transmission the proportionally largest delays are experienced within the network. They can be divided into fixed amounts of propagation delays on the network trunks and variable queuing delays inside routers and other equipment. As a rule of thumb $4 - 6\mu s/km$ network delay can be assumed on WAN links [69]. Since speech coders mostly provide constant bit-rate streams, usually a de-jitter buffer is used to compensate the variable delay portions at the

receiver side. The de-jitter buffer additionally introduces a fixed amount of delay. It holds received samples for a period of time in order to provide a fixed-delay stream to the voice decoder. There exist also some implementations of adaptive de-jitter buffers, but this assumption might not be useful when assesing VoIP quality. We are actually more interested on upper-bounds of the total delays. A summary of the previously mentioned delay portions and the locations of their occurrence is given in Fig. 10.

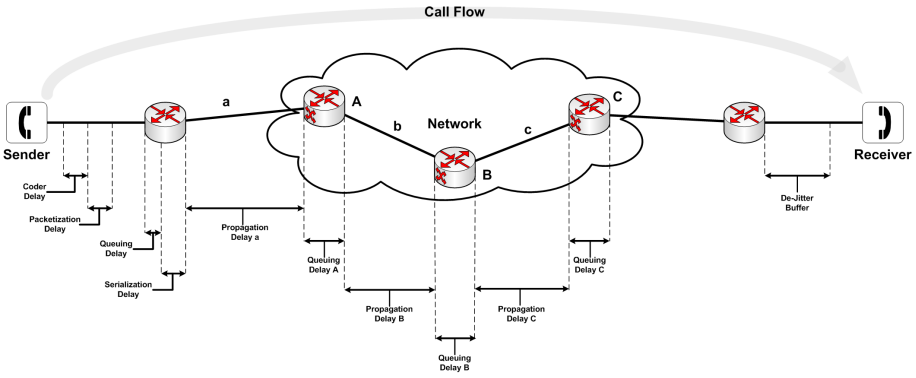


Figure 10: VoIP Delay Portions.

In order to estimate the voice quality by the means of ITU G.114 we further introduce a VoIP Delay Budget separated by fixed and variable delay portions as stated above and values derived in accordance with [22]. For this purpose we define some new metrics below and give the Delay Budget in alignment with our setup in Table 3.

Inter-Packet-Delay – $IPD_{i,\delta}$ between packets p_i and p_{i+1} is the time interval between both packets, either the desired one before sending, indicated by $\delta = Des$, or the one at transmission, indicated by $\delta = Tx$, or the one at receiption, indicated by $\delta = Rx$. Therefore *Inter-Packet-Delay* becomes

$$IPD_{i,\delta} = \begin{cases} t_{i+1,Des} - t_{i,Des}, & \delta = Des \\ t_{i+1,Tx} - t_{i,Tx}, & \delta = Tx \\ t_{i+1,Rx} - t_{i,Rx}, & \delta = Rx \end{cases} \qquad 1 \leq i \leq n-1. \tag{15}$$

Scheduling-Delay-Variation – SDV_i of voice packet p_i is the deviation of the desired *Inter-Packet-Delay*, $IPD_{i,Des}$ and the actual *Inter-Packet-Delay* at transmission, $IPD_{i,Tx}$. Since we want to avoid error-prone handling with absolute time values, we use relative times instead and only define $(n-1)$ values, i.e.

$$SDV_i = \{t_{i+1,Tx} - t_{i,Tx}\} - \{t_{i+1,Des} - t_{i,Des}\} \quad 1 \leq i \leq n-1. \tag{16}$$

Reading Table 3 furthermore provides us the ear-to-mouth delay EMD_i of the $i-th$ packet p_i depending on the chosen de-jitter buffer size, which we refer to as $DJBS$. To be more formal, we define:

Ear-To-Mouth-Delay – EMD_i of voice packet p_i is calculated as the time the packet needs to be generated (Coder Delay+Packetization Delay), to be transmitted (Scheduling-Delay-Variation, SDV_i), to be received (One-way-Delay, OWD_i), and to be delivered to the voice decoder (De-Jitter-Buffer-Delay, $DJBS$), i.e.

$$EMD_i = 10\ ms + 20\ ms + SDV_i + OWD_i + DJBS \quad 1 \leq i \leq n-1. \quad (17)$$

Delay Type	Fixed Delay Portion	Variable Delay Portion
Coder Delay	10 ms	–
Packetization Delay	20 ms	–
Queuing Delay	–	incl. in SDV
Serialization Delay	incl. in OWD	–
Network Delay	–	incl. in OWD
De-Jitter Buffer Delay	40-120 ms	–
Total Delay	70-150 ms + OWD + SDV	

Table 3: VoIP Delay Budget.

We now have the needed metrics to determine the delay range of a given voice stream within the G.114 recommendation. Though, what still remains is an appropriate model for assesing the packet-loss. It should be noted that packet-loss does not only occur as a result of undelivered packets from the network but also in consequence of discarded late packets in the de-jitter buffer. We therefore adopt a simple, non-adaptive de-jitter buffer model from [10] and assume a static de-jitter buffer with, a total size of $2br$ in average delay terms, where $2b$ defines the total number of voice packets that could be held in the buffer and r denotes the expected inter-packet arrival period, i.e. the packetization interval of the vocoder. Let us further assume that the first packet in a call is buffered until the bth packet arrives and play-out starts with the arrival of packet $b+1$. Additionally our buffer shall be able to hold early packets but should discard late packets, i.e. the latter contribute to packet-loss. A late packet p_i then would be discarded if its Inter-Packet-Delay at receiption, $IPD_{i,Rx}$, exceeds half the de-jitter buffer size br. Hence, we define

De-Jitter-Buffer-Packet-Loss – $DJBPL_i$ is counted as one if packet p_i was late, or it is counted as zero if p_i is received in time, i.e.

$$DJBPL_i = \begin{cases} 0, & IPD_{i,Rx} \leq br \\ 1, & IPD_{i,Rx} > br \end{cases} \quad 1 \leq i \leq n-1. \quad (18)$$

To determine the overall One-way packet-loss $OOPL_{i,total}$ to be used for assessing the VoIP quality according to Fig. 9 we finally sum-up the *UDP-One-way-Packet-Loss* and the *De-Jitter-Buffer-Packet-Loss*, therefore

$$OOPL_{i,total} = OPL_i + DJBPL_i \quad 1 \leq i \leq n-1. \tag{19}$$

A further step taking into account burstiness of packet-loss and delay would be the definition of a service outage probability. But this remains future work. The basic idea however is to calculate the ratio of measured ear-to-mouth delays and packet-losses within a defined service class area in Fig. 9 and values lying outside. The proportion then would give the service outage probability. Fig. 11 illustrates the explained mechanism.

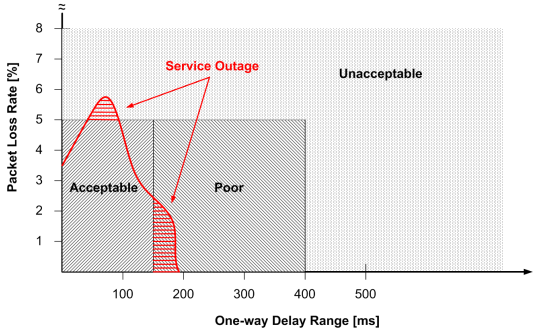


Figure 11: VoIP Service Outage Probability Metric.

3.2.4 Toolchain Validation

In the previous Sections we considered partial aspects of data aquisition, probe design and evaluation of measurement data. But how to validate the entire toolchain in order to detect implementation or systematic errors? A suitable and convenient approach is to test the toolchain in a well-controlled environment with predefined characteristics. These properties can e.g. be provided by a network emulator. To validate our measurement toolchain we use Linux' built-in network emulator *NetEm* [19]. It provides packet filtering on interface outputs and can be easily configured through a command line utility. *NetEm* itself is a small kernel module which interacts with the queuing discipline of the network stack. The default Linux network queuing discipline is FIFO which can be altered with the help of the *NetEm* command line utility to emulate WAN links. Currently the emulation of (variable) delay, loss, duplication and re-ordering is supported. For the purpose of validating our tools we emulate normally distributed delays with a given standard deviation as well as packet loss on the host that serves as sender. The probe traffic then is captured on the receiver side and after evaluation of the data we should have measured the predefined delay distribution and packet-loss respectively.

4 Measurement Results

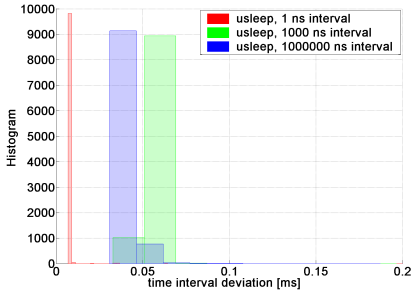
In order to investigate the relations of path characterstics and the number of traversed hops in the Internet we performed measurements using various selected PlanetLab nodes across the globe. We first obtained extensive topology information in two traceroute-based campaigns. During January-February 2009 we examined more than 500 PlanetLab hosts targeted from our node in Berlin. The gathered topology data thereafter was used to select further 20 nodes for one-way measurements. Within these measurements we collected 100,000 samples each time from Poisson-distributed UDP probe streams and evaluated several path characteristics. Additionally we performed Voice over IP and Available Bandwidth measurements on a few routes in order to assess the effects of these path characteristics on real-world application performance. Since one-way measurements heavily depend on time precisions we thouroughly checked clock accuracies and resolutions in an further effort. In this Section we first validate our measurement approach and go on with the investigation of the PlanetLab topology. Following we present the evaluated one-way path characteristics and finally show results of Voice over IP performance and Available Bandwidth estimation.

4.1 Validation and Clock Issues

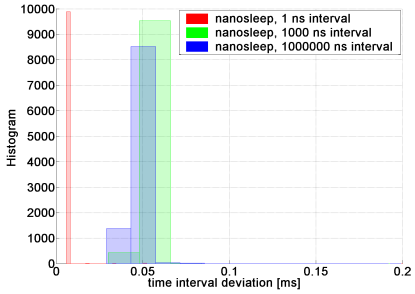
For an accurate error estimation within our measurements we investigated the clock resolutions including process scheduler delays on a reference system and our PlanetLab node. The reference system, which was an old Pentium III machine in our lab, was equipped with a PIT clock and the deployed Linux kernel was *hrtimers*-enabled and patched for real-time preemption. The PIT clock showed a minimum resolution of $12.5\mu s$ through the `cat/proc/timer_list` interface. We tested three timer intervals from $1ns$ to $1s$ as well as the `usleep()` (Fig. 12a) and `nanosleep()` (Fig. 12b) timers which we use in our probe engine to schedule packets. The actual time values were obtained through the `gettimeofday()` function. Both timers achieved a mean accuracy of $50\mu s$ at $1\mu s$ and $1s$ time intervals respectively. The standard deviation was at $\sigma \approx 50\mu s$, too. Using $1ns$ intervals both timers even reached a $5\mu s$ accuracy with a standard deviation of $\sigma \approx 5\mu s$ as well. We believe that this different behaviour in the long interval regime is a result of clock counter overflows. Since we only used 10,000 samples (successive time intervals) in each measurement the counter did not overflow when using rather short $1ns$ intervals. Further on, we note that all values are obtained under low system load.

Now having reference values for an optimized system we performed the same measurements on our PlanetLab node. The `cat/proc/timer_list` interface states a $1.0\mu s$ resolution by using a LAPIC clock. The `usleep()` timer (Fig. 13a) and likewise the `nanosleep()` timer (Fig. 13b) show a quite homogeneous behaviour. The mean accuracy in all cases amounts to $1ms$ with a standard deviation of $\sigma \approx 150\mu s$. This result is rather disappointing compared to the reference

system. For our network measurements this means in consequence that recorded timestamps can only be treated to be accurate within in a range of about $\pm 1ms$. We assume that the higher resolution of the process scheduler is caused by the underlying *vserver* resource management but we could not obtain detailed information on that mechanism so far. However, we now have a lower bound reference to asses the *Tx-Host-Noise* in our measurements.

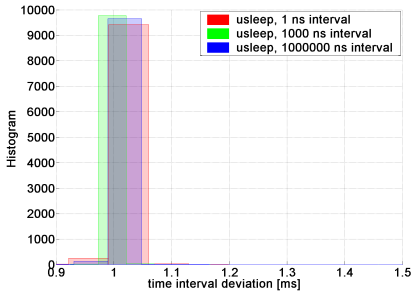


(a) usleep(). Mean deviation from desired timer interval $50\mu s$, $\sigma \approx 50\mu s$.

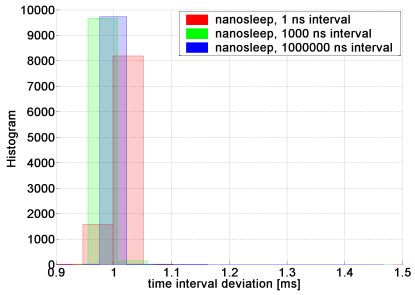


(b) nanosleep(). Mean deviation from desired timer interval $50\mu s$, $\sigma \approx 50\mu s$.

Figure 12: Reference System Timer Accuracy.



(a) usleep(). Mean deviation from desired timer interval $1ms$, $\sigma \approx 150\mu s$.

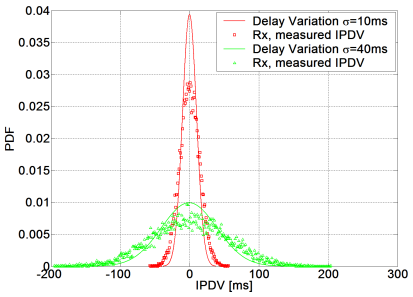


(b) nanosleep(). Mean deviation from desired timer interval $1ms$, $\sigma \approx 150\mu s$.

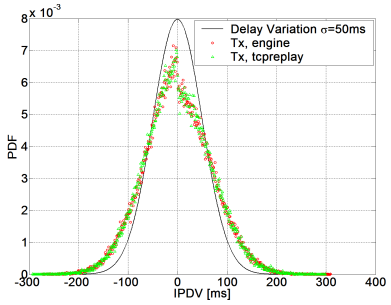
Figure 13: PlanetLab Timer Accuracy.

The next step in the validation process is the verification of our measurement approach. For this purpose we performed measurements inside a LAN while emulating WAN conditions. As mentioned earlier we utilized Linux *NetEm* to generate normally distributed delays in the outgoing queue of the sender host. We reused our prementioned reference system as the sender and a similar system as the receiver. Our probe engine then was installed on both systems for verification. Two measurements with a predefined delay standard deviation of $\sigma = 10ms$ and $\sigma = 40ms$ respectively were used to calculate the received IPDV from the recorded

data. The probe stream was sent at a $1,000ms^{-1}$ Poisson average-rate and 10,000 packets were transmitted and received each time. In the case of $\sigma = 10ms$ an IPDV with a standard deviation of $\sigma_{IPDV} = 14.3ms$ was determined as well as an IPDV with $\sigma_{IPDV} = 54.7ms$ in the case of $\sigma = 40ms$ (Fig. 14a). In order to assess the performance of our packet-player with a sophisticated existing solution we also compared our engine against tcpreplay. For this purpose we evaluated the Tx-Host-Noise which in fact corresponds to the NetEm delay distribution in this case (due to the working mechanism of NetEm). Both tools were used to transmit a 10,000 packet probe stream as stated above and showed identical performance. While NetEm was configured for a delay distribution with $\sigma = 50ms$, tcpreplay produced a Tx-Host-Noise with $\sigma_{tcpreplay} = 66.6ms$ and our tool with $\sigma_{engine} = 66.9ms$ respectively (Fig. 14b). Taking into account that we cannot measure the accuracy of the NetEm-induced delay we suppose the above results to be sufficient with respect to our demands.



(a) Measured IPDV (squares and triangles) with emulated delay distribution of $\sigma = 10ms$ and $\sigma = 40ms$ (solid lines).



(b) Emulated delay distribution of $\sigma = 50ms$ (solid line) and Tx-Host-Noise measured with our Probe Engine and Tcpreplay.

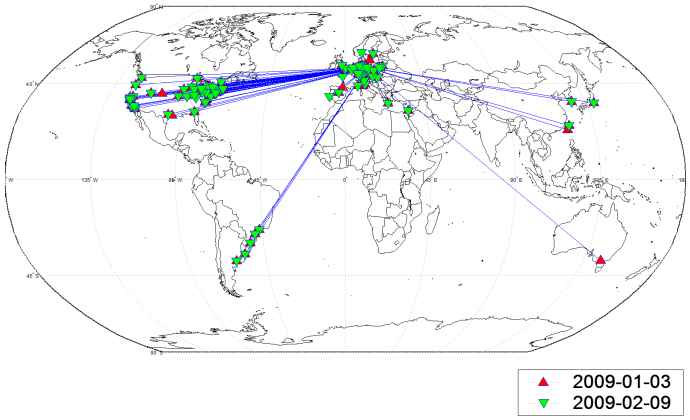
Figure 14: Toolchain Validation with Network Emulator.

4.2 PlanetLab Survey and Round-trip Path Characteristics

Our PlanetLab round-trip path characteristics and topology discovery campaign comprised two *traceroute*-based measurements. We gathered one dataset in the beginning of January 2009 (2009-01-03) and another one in the middle of February (2009-02-09). All considered nodes were first filtered through the *CoMon* monitoring site and targeted, only if they were in "alive, no problems" state. This procedure prevents from selecting faulty nodes exhibiting various problems, such as heavy clock drifting, low disk spaces, or DNS problems (see [44]). Since we were also interested in geographical positions, node coordinates were further determined through automated *hostip.info* database queries in a parallel process. The obtained data also served as groundwork for a further selection procedure in order to specify suitable nodes for one-way path characteristics measurements. Nonetheless, various hop-count specific coherences could be observed within the data.

4.2.1 Topology and Traceroute Statistics

Currently *planetlab.org* states 996 nodes distributed over 485 sites while 591 of the nodes participate in the CoMon monitoring (only 60%, as of March 2009). In the first measurement we found 508 nodes properly operational which is only 86% of all monitored hosts and only 235 could be geo-located, which amounts to only 40% of the CoMon list. The second measurement revealed an operational availability of about 89% (527 nodes) and 43% (255 nodes) were locateable. The unavailable nodes were probably either down or did not respond to ICMP messages. Also packet-loss may be a possible reason. Fig. 15 shows the determined node positions in both measurements. From that we observe e.g. that Melbourne was not reachable in 2009-02-09, which we fittingly found to suffer from heavy packet-loss later on. We further discover that most PlanetLab nodes are clustered in Europe and the US while a significantly smaller percentage is distributed over South America, the Arabian Peninsula, Asia-Pacific and Australia.

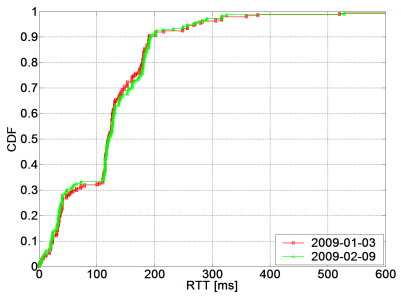


(a) 2009-01-03, 16:55 CET; 235 of 508 operational nodes were located.
2009-02-09, 23:35 CET; 255 of 527 operational nodes were located.

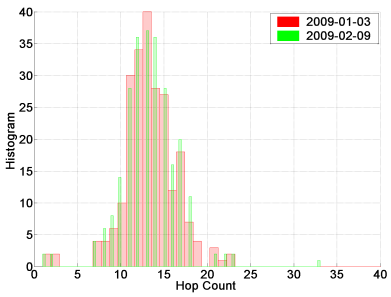
Figure 15: PlanetLab Geotargeting Results.

Originated from our node in Berlin (*planetlab01.tkn.tu-berlin.de*) all measured (and worldwide distributed) locations were reachable within $133ms$ ms (RTT) in average via the Deutsches Forschungsnetz (DFN) [12] network which our node connected to. The RTT distributions of both measurements are shown in Fig. 16a. From that we recognize the previously mentioned clustering of PlanetLab nodes. Europe and Arabian Peninsula lie within a range of $100ms$ RTT, the US east coast within $140ms$, US west coast and (partly) Asia-Pacific within $200ms$ and South America, (partly) Asia-Pacific as well as Australia are reachable in $200ms$ RTT and above. The maximum response times of around $2,000ms$ likely occured from Australia but were also observed from some hosts in Europe. Australia however occasionally showed observed RTT values of over $12,500ms$ yet.

The number of traversed hops indeed seems to be normally distributed as shown in Fig. 16b. All determined routes took about 13 intermediate paths in average until the destination hosts were reached. The maximum Hop-Count was observed for *planetlab1.eurecom.fr* and totals to a measured quantity of 107 hops. We assume this value to be provoked by a routing-loop or similar effects. Later measurements unveiled a more expextably Hop-Count of 16 intermediate paths for this node. This example imposingly shows the need for more sophisticated traceroute mechanisms to be aware of a variety of routing effects like load-balancing and other anomalies.



(a) Round Trip Times. 2009-01-03: max. 1875.2 ms, mean 133.43 ms (red); 2009-02-09: max. 2251.6 ms, mean 127.2 ms (green).



(b) Hop Counts. 2009-01-03: max. 107 Hops, mean 13.67 Hops (red); 2009-02-09: max. 33 Hops, mean 13.32 Hops (green).

Figure 16: PlanetLab Traceroute Statistics.

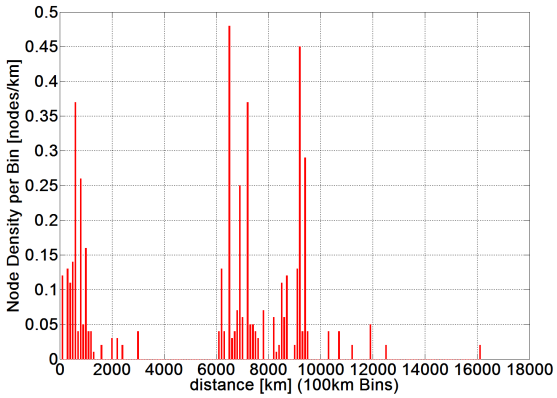
4.2.2 Distance Related Results

Based on the resolved host coordinates, we further looked at characteristics in relation to geographic distances of PlanetLab nodes. Originating again from our node in Berlin we calculated the corresponding airline distances per node using (7), and from that an estimate of the appropriate fiberline distances applying (8).

As expected, Fig. 18a shows that the fiberline distances are noticeably higher compared to the airline distances. Moreover, they significantly dissociate by a clearance of $1,500km$ from a range of about $6,000km$ (transatlantic connections) and onwards. The maximum computed airline distance was $15,977km$, which coincides well with the distance Berlin-Melbourne. We also notice that all distances follow the same distribution as the RTTs, previously shown in Fig. 16a. Therewith we can equally match several distance steppings to the geographical clustering of PlanetLab nodes. We identified sites located in Europe and the Arabian Peninsula within a range of $3,000km$. From $6,000km$ to $7,000km$ we expect sites on the US east coast and in the segment up to $9,500km$ sites on the US west coast and Asia-Pacific. The upper segment, above $10,000km$, contains all other sites located in South America and Australia.

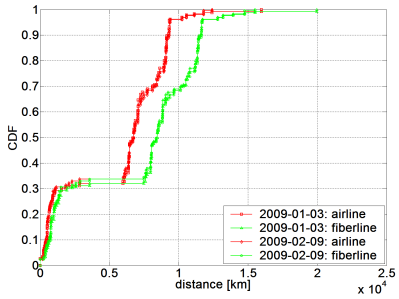
For a further analysis of spatial distributions of PlanetLab nodes, we merged the data from both measurements and determinded the number of nodes located within $100km$ distance steps starting from Berlin, i.e. we obtained densities of nodes reachable within designated $100km$-wide circular areas around Berlin. As we are interested in spatial characteristics at this point, airline-distances were used in the following. Therewith, the observed spatial densities of PlanetLab nodes are shown in Fig.17. From that we found a moderate amount of 161 nodes deployed within a range of $3,000km$ (Europe), the majority of 310 nodes deployed between $6,000km$ and $9,500km$ (US and Asia-Pacific), and a rather low amount of only 19 nodes for distances above $10,000km$ (South America/Australia). We further note, that in the case of $6,000 - 9,500km$-distant areas, the majority of nodes are located in the US rather than in Asia-Pacific regions (see Fig. 15). However, while the average distance between nodes in Europe was $683km$ and $1,713km$ in the US/Asia-Pacific areas, the mean node density per $100km$ is still higher for the latter one. Statistically we therefore would observe around $9nodes/100km$ in the US and $5nodes/100km$ across Europe. In the case of the South-American and Australian area the node deployment density would even fall to comparatively low $0.3nodes/100km$.

Hence, we notice from the findings above, that inferences drawn from our *traceroute* measurements will most probably apply to connections between Europe and the US only, since the vast majority of resolved PlanetLab nodes is deployed in these regions.

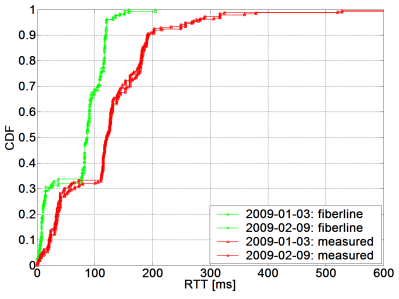


(a) Europe/Arabian Peninsula: $5nodes/100km$, US/Asia-Pacific: $9nodes/100km$, South-America/Australia: $0.3nodes/100km$.

Figure 17: PlanetLab Spatial Node Density.



(a) Estimated airline & fiberline distances of sites within PlanetLab. $0 - 3,000km$ Europe; $6,000 - 7,000km$ US east coast; $7,000 - 9,500km$ US west coast and Asia-Pacific; $\geq 10,000km$ South America and Australia.



(b) Estimated fiberline RTT distribution of sites within PlanetLab compared to measured RTT distribution.

Figure 18: PlanetLab Distance Statistics.

In the next step we used the calculated fiberline-distances instaed of airline-distances in order to estimate packet propagation delays due to optical path lengths between nodes. From that we investigated differences of observed RTTs and corresponding inter-node propagation times.

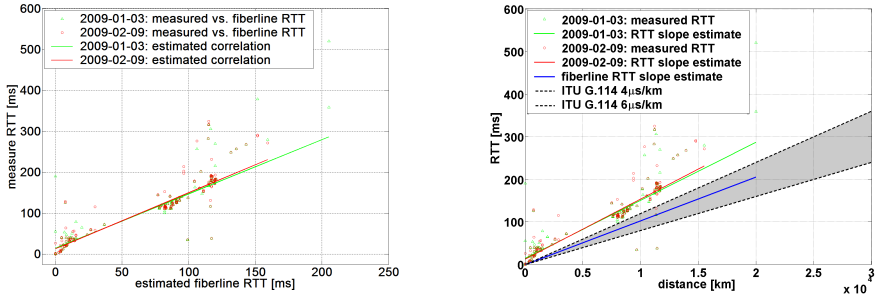
In optical fibers the speed of light depends on the refractive index n of the used material. The refractive index itself is calculated from the group index which further depends on the wavelength. The light then travels at a speed c_{fiber} along the optical cable whereas c_{fiber} is proportional to the cable's refractive index. The speed within the medium by then is given through

$$c_{fiber} = \frac{c_0}{n}, \tag{20}$$

with $c_0 = 299,792,458m/s$ the speed of light in the vacuum. Typical values of refractive indices within optical fibers are in the range of 1.46 to 1.54. We pick $n = 1.54$ as an upper bound and use $c_{fiber} = 194,670,427m/s$ to calculate an estimated RTT out of twice the obtained fiberline distances, i.e. we assume packets to take the same forward and return route under nearly equal load conditions. We compared the estimated fiberline RTTs with our measured RTTs in Fig. 18b and notice that both distributions roughly exhibit equal shapes. This is not surprising since we already recognized that the calculated distances conform with the measured RTTs in their distribution. But we make the finding that measured RTTs (RTT_{meas}) averagely seem 1.5 times as large as the corresponding fiberline RTTs (RTT_{fiber}). Applying robust line fitting techniques we further compared the measured RTTs against estimated RTTs and ratified this expectation in Fig. 19a. Robust line fitting thereby is less sensitive to outliers in the data, which we slightly observed in our measurements. So we list this finding by

$$RTT_{meas} \approx 1.5 \cdot RTT_{fiber}. \tag{21}$$

In order to verify (21) we further assessed our RTT estimation approach by comparison with the ITU G.114 assumption of $4-6\mu s/km$ delay in Wide Area Networks (also see Section 3.2.3). From Fig. 19b we obtain a slope of $5.14\mu s/km$ delay for our estimated fiberline-distance RTTs which lies exactly in between the range of the proposed values from ITU G.114. Further on, for the measured RTTs we find a slope of $7.5\mu s/km$ which roughly gives 1.5 times the fiberline delay as claimed in (21). Considering that the ITU provides delay values for signal propagation in transmission media only, we assume that our estimate is quite applicable. Thereof we further conclude that approximately one third of the measured delays must be originated from queuing delays in routers.



(a) Estimated fiberline RTTs vs. measured RTTs. Correlation goes to: $RTT_{meas} \approx 1.5 \cdot RTT_{fiber}$.

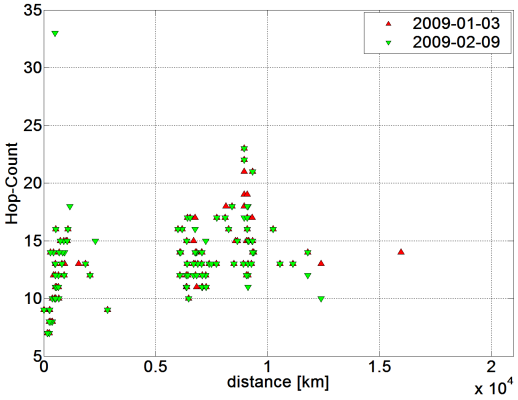
(b) Delay slopes. Est. (blue; $5.14\mu s/km$); meas. (red, green; $7.5\mu s/km$); ITU G.114 (gray; $4-6\mu s/km$).

Figure 19: PlanetLab RTT Spatial Correlations.

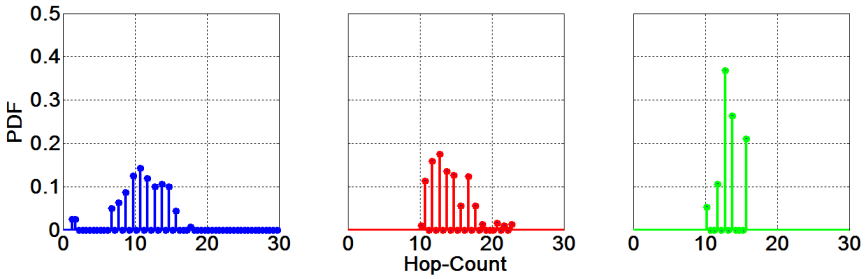
4.2.3 Hop Count Related Results

In the previous Section we found based on measured correlations of distances and delay, that on particular routes almost one third of the packet delays must be induced by queuing latencies in routers. Since packets mostly traverse multiple routers, the number of intermediate hops should also have an impact on the overall delay. In this Section we therefore reuse computed fiberline and airline distances from above, and therewith investigate the relation of the number of traversed hops and delay. In order to complete our observations, we further examine geographical distributions of deployed routers as well.

To get started, we investigated relations of Hop-Counts and their corresponding distance dependencies. Comparing the calculated airline distances as well as measured Hop-Counts, we observe from Fig. 20a the already noticed clustering of values within mainly three segments. This likewise applies to Hop-Counts and distances. Again, for the first segment within a range of $3,000km$ we expect nodes located in Europe and the Arabian Peninsula. From about $6,000km$ to $9,500km$



(a) Hops vs. Airline-Distance. Clustering between: $0-3,000km$, $6,000-9,500km$, $10,000-16,000km$.



(b) Hop-Count Distribution per Cluster. Clusters: left: $0-3,000km$, middle: $6,000-9,500km$, right: $10,000-16,000km$.

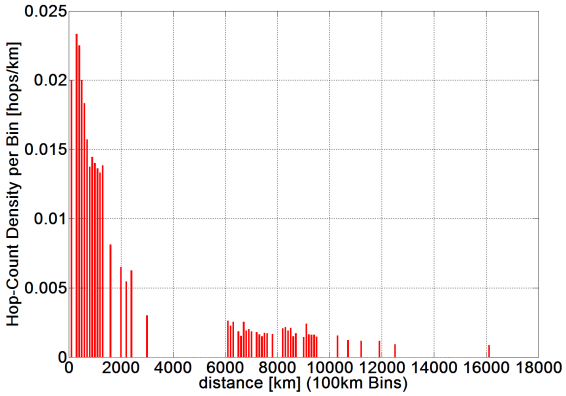
Figure 20: PlanetLab Hop-Count & Distance Statistics.

we found nodes from the US as well as Asia-Pacific and in the last segment with distances above $10,000km$, nodes from South-America and Australia. Since both *traceroute* measurements delivered comparable results, we merged all obtained values in the following to gain a broader spectrum of data. Furthermore, the merged data then was disassembled on a per-distance basis, which allowed us to examine Hop-Count and distance correlations for each cluster separately.

Therefore we obtain from Fig. 20b roughly normally distributed $1-18$ hops in the European cluster with mean Hop-Count of 11. Further on, $10-23$ hops with mean Hop-Count of 14 in the US and Asia-Pacific cluster, and finally $10-16$ hops and also with mean Hop-Count of 14 for the long-distance, i.e. South-American and Australian areas. This shows, that most intercontinental destinations, independently of particular distances between source and target hosts, are reachable within the same range of traversed routers. Obviously, we further notice the highest spread of routers across Europe and the US, which however is most probably caused by the comparatively large quantities of PlanetLab nodes installed on both continents, therewith exploiting a higher variety of possible paths towards target nodes.

Next, we examined the spatial density of Hop-Counts per $100km$ distance steps, using the same approach as described in Section 4.2.2 before. From Fig. 21 we therewith observe an exponentially decreasing density of traversed routers with increasing distance. The highest Hop-Count density thereby occurs within the first $300km$, which we suppose to be caused by a relative high number of routers throughout the access network. The density of routers herein amounts to approximately $2hops/100km$. Further on, the router density then rapidly decreases for distances up to $800km$ and remains relatively constant throughout other Central European regions, i.e. for distances up to $1,500km$. The overall Hop-Count density within Europe, however, can approximately be quantified to $1hop/200km$. Across the US on the other hand, we recognize a considerably smaller and rather homogeneous density of traversed routers. Statistically we would observe about $1hop/800km$, which is only one fourth of the Hop-Count density throughout Europe. Note that this value is not necessarily the same for paths inside the US, but rather determines characteristics of specific routes for traffic directed from the US east coast to the US west coast. Finally, we further notice router densities for distances above $10,000km$ also being nearly equally distributed, but since nodes are widely scattered in this regions and their densities are rather low, a meaningful Hop-Count density cannot be inferred from the present data.

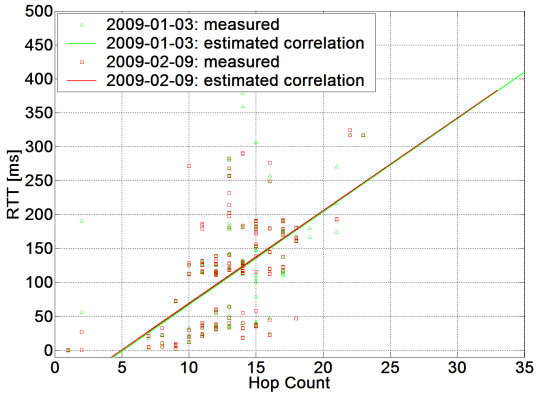
In summary, assuming that the number of traversed routers significantly impacts the experienced overall packet delay and considering the observed router densities as shown above, we would expect overall packet delays growing considerably faster compared to optical path delays in the European area and only slightly diverging on routes across the US. Fig. 18b from Section 4.2.2 above confirms this expectation to a high propability.



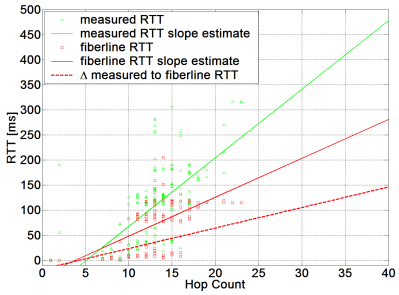
(a) Europe $(0-3,000km)$: $\approx 1-2hops/100km$, US: $(6,000-9,500km)$: $\approx 1hop/800km$.

Figure 21: PlanetLab Spatial Hop-Count Density.

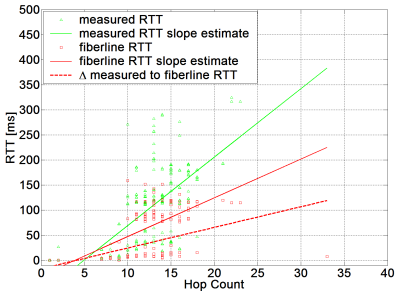
In order to quantify the expected Hop-Count dependent delay increase more precisely, we estimated $RTT/2$ per hop in order to obtain the occurring delays if a particular number of routers is traversed. For that reason we used robust line fitting and thereof recognize from Fig. 22a the $RTT/2$ of about $6.5ms/hop$, i.e. each traversed router is expected to add $6.5ms$ to the overall packet delay. Since this value is obtained from measured RTTs, it still includes the fiberline delays, meaning delay portions evoked by signal propagation through optical fibers. Consequently we reduced the measured per-hop RTTs by the fraction of corresponding propagation delays (denoted as Δ) as shown in Fig. 22b and 22c. From that, we further observe at least a slope of $2.0ms/hop$, i.e. this value represents the time interval each packet is averagely delayed within particular router queues.



(a) Measured RTT vs. Hops. Slope: $\approx 6.5ms/hop$.



(b) 2009-01-03: RTT vs. Hops. Estimated slope: $2.0ms/hop$ (red).



(c) 2009-02-09: RTT vs. Hops. Estimated slope: $2.0ms/hop$ (red).

Figure 22: PlanetLab RTT Hop-Count Correlations.

4.3 One-way Path Characteristics

Based on topology information gathered within our *traceroute* geo-targeting measurements (see Section 4.2), we selected 20 nodes for further investigations of one-way path characteristics (packet dynamics) dependent on the number of traversed hops. All measurements were performed on routes originating from our node in Berlin (*planetlab01.tkn.tu-berlin.de*) and ceasing at the corresponding target node. All destined nodes were selected in a manner that they cover a wide range of $(6-18)$ hops and various geographical locations (see Fig. 23). Since one-way performance measurements heavily depend on proper synchronization between hosts, we also noticed to omit "drifting nodes" as stated by the *CoMon* site, i.e. nodes diverging more than 1 minute from the median PlanetLab time (see [44]). Moreover, to retain comparability between obtained results, all measurements were conducted using identical probe stream setups as listed in Table 4 and, where possible, multiple nodes exhibiting equal hop-counts were selected to gain better diversity. Also note, that Poisson distributions were generated independently for each measurement run. Finally, all data were collected between February 11th to 15th and particular target nodes, as well as corresponding hop-counts (HC), node locations (Loc.), and measurement initiation times are given in Table 5.

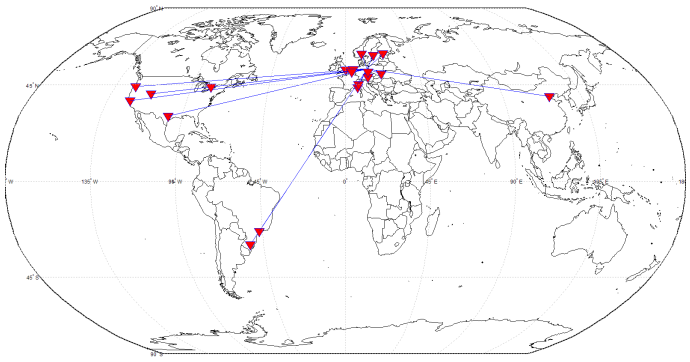


Figure 23: One-Way Path Characteristics. Target Node Locations.

Parameter	Value
Packet Type	$40Byte$ UDP/IP
Number Packets	100,000
Poisson Packet Rate	$100ms^{-1}$
Duration	$\approx 168min$

Table 4: One-way Path Characteristics. Probe Stream Setup.

Target Node	HC	Loc.	Date/Time (UTC)
dschinni.planetlab.extranet.uni-passau.de	6	DE	11.02.2009/21:30:22
planet2.inf.tu-dresden.de	6	DE	11.02.2009/23:29:18
pl1.bit.uoit.ca	9	CA	12.02.2009/11:29:20
planck227.test.ibbt.be	9	BE	12.02.2009/10:29:25
planetlab2.cs.vu.nl	10	NL	12.02.2009/22:29:28
planetlab1.pop-rs.rnp.br	12	BR	13.02.2009/02:45:47
planetlab2.cs.uoregon.edu	12	US	12.02.2009/08:29:56
planetlab2.di.unito.it	12	IT	12.02.2009/07:29:46
planetlab2.ifi.uio.no	12	NO	12.02.2009/06:29:38
planetlab2.s3.kth.se	12	SE	13.02.2009/03:45:38
planetlab3.hiit.fi	12	FI	12.02.2009/04:29:45
planetlab4.mini.pw.edu.pl	12	PL	12.02.2009/03:29:58
planet2.scs.stanford.edu	14	US	12.02.2009/17:29:58
planetlab2.csres.utexas.edu	14	US	12.02.2009/16:30:09
planetlab3.xeno.cl.cam.ac.uk	14	GB	12.02.2009/23:30:17
planetlab2.eurecom.fr	15	FR	12.02.2009/02:30:10
planetlab2.larc.usp.br	15	BR	12.02.2009/01:30:01
planetlab3.flux.utah.edu	17	US	15.02.2009/14:12:31
pl-node-0.csl.sri.com	17	US	15.02.2009/15:08:01
planetlab1.iin-bit.com.cn	18	CN	12.02.2009/12:30:28

Table 5: One-way Path Characteristics. Measurements Schedule.

4.3.1 One-way Delay

Examining the captured probe packets at sender and receiver we obtained the One-way Delay (OWD) as described in Section 3.2.1. Out of the couple of investigated path characteristics, OWD is the most critical one, since its accuracy primarily relies on proper synchronization, low clock skew and drift. Moreover, observed OWDs can highly vary under different load conditions across the network path under investigation.

Fig. 24 shows the cumulative probability distributions (CDFs) of measured OWDs on all 20 paths while the coloring indicates the particular Hop-Count towards the target node (note, that we will retain this coloring scheme within all figures throughout the rest of the Section). From that we recognize OWD values in the range of $5 - 225ms$ over all Hop-Count values. Since PlanetLab nodes are not synchronized via GPS or an equivalent precise time source, the observed delays remain somewhat unreliable. Comparisons with additionally measured Round Trip Delays ($RTDs$) however, showed the OWDs in good agreement with $RTD/2$ in most cases. This indicates that the pre-selection of "drifting nodes" via the *CoMon* site is

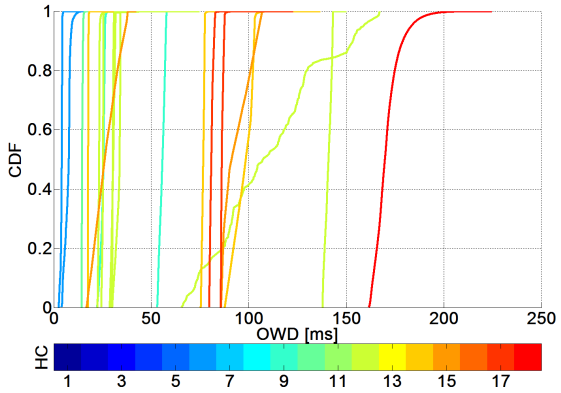


Figure 24: Overall OWD Statistics.

suitable to eliminate hosts exhibiting high relative Offsets. Nonetheless, we further notice from Fig. 24 some (at least 5) hosts revealing a high monotonic increase of OWD with likewise increased cumulative probability. While we already mentioned that network load conditions affect the measured values of OWD, steadily raising network traffic leading to permanently increased OWDs seems quite uncommon. Instead, we assume this behaviour more likely evoked by the presence of relative Skew and Drift between nodes. If sender and receiver clocks were perfectly running at same frequencies, i.e. no relative Skew is present, and load conditions in the network were at virtually constant low levels, we would expect the CDFs on particular paths becoming straight vertical lines. Further on, additionally assuming highly variable load conditions on the other hand, we would expect OWDs exhibiting some randomness, leading to longer and smooth tail probabilities, as is the case for the (red) rightmost OWD curve, observed on the route to *planetlab1.iin-bit.com.cn* (note, that Skew is still present here).

Fig. 25 therefore depicts measured OWDs of the five ”most drifting” nodes, illustrating the impact of different clock anomalies (relative Skew and Drift) over the whole observation time. We notice (i) linearly decreasing OWDs (measured at *planetlab2.eurecom.fr*), indicating a receiving node's clock slower than the sender's clock, (ii) piecewise linearly increasing OWDs (*planet2.scs.stanford.edu*, *planetlab2.larc.usp.br*), indicating receiving node clocks faster than sender node clocks and most probable NTP-induced clock frequency shifts, (iii) linearly decreasing OWDs overlaid by heavy (network-induced) noise (*planetlab1.iin-bit.com.cn*), and (iv) purely random and highly variable behaviour (*planetlab2.cs.uoregon.edu*), delineating it nearly infeasible to distinguish between clock anomalies and network-induced distortions. In order to verify that the shown OWD behaviour is inherited by clock anomalies, we checked the estimated skew progressions against the recorded drift values within *CoMon*, which showed similar behaviour and therewith confirm that expectation. While we would suppose (i)-(iii) to be correctable

to a sufficient degree through post-processing with appropriate clock skew removal techniques (see Section 2.5.5), this remains future work. Hence, as we recognize relative Skew and Drift having a substantial impact on precise OWD measurements (more than $100ms$ dynamic range in case (iv)), we omit the five named "most drifting" nodes for further investigations and suppose the remaining ("low-skew") node's OWDs to be relatively accurate. However, we note that these OWDs still contain uncertainties induced by clock anomalies.

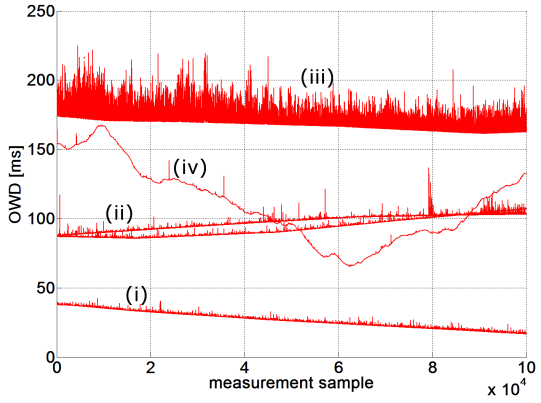
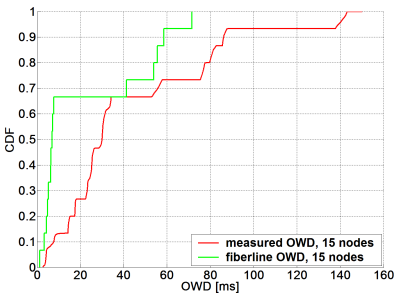


Figure 25: OWD Clock Skew and Drift.

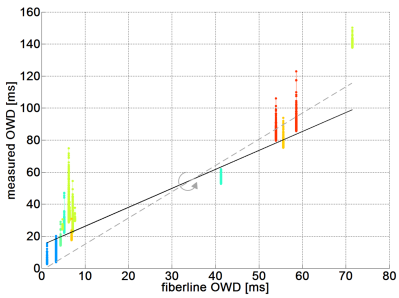
However, keeping unreliabilities in mind, we calculated each node's fiberline path length and therefrom corresponding path delays using again (7), (8) and (20) respectively, and investigated the correlation of measured OWDs (OWD_{meas}) against pure signal propagation times (OWD_{fiber}) in analogy to Section 4.2.2. Therewith, applying robust line fitting on all values, we observe from Fig. 26b a ratio of approximately $OWD_{meas} \approx 1.38 \cdot OWD_{fiber}$ (solid line), but also recognize a shift of the regression line in the axis origin of about $14ms$. We expect at least three main reasons causing that behaviour. First, we assume OWDs increasing faster in low-distance regions, owed to stronger influences of Hop-Count related (queuing) delays, which is in contrast to long-distance paths, that stronger depend on the impact of pure signal propagation delays. This in fact causes a non-linear relation between both values, not reproduced by the line fitting algorithm. Fig. 26a, further showing the CDFs of measured as well as fiberline OWDs, clearly illustrates that issue. Second, we notice a strong clustering of values around $OWD_{meas} = 30ms$ and only a few results in the delay regime below. The lack of sufficient observations in the latter range therefore additionally causes a pull-up of the regression line. Last but not least, OWD variability and clock Skew also lead to an overall shift of measured values. Nonetheless, assuming absence of the three mentioned uncertainties, we would expect the regression line being wrenched around its median, such that it comes closer to the axis origin in the low-delay regime and being pulled up in the high-delay regime. In order to estimate an upper

bound for the ratio between OWD_{meas} and OWD_{fiber} in this case, we substract the $14ms$ offset in the axis origin and add them to the maximum of the regression line, resulting in $OWD_{meas} \approx 1.58 \cdot OWD_{fiber}$ (dashed line). Since we already observed a relation of $RTT_{meas} \approx 1.5 \cdot RTT_{fiber}$ from our *traceroute* measurements (see Section 4.2.2), we likewise suppose this ratio for OWDs lying in between the estimeated 1.38 and 1.58. Hence, the expectation of one third of packet delays being induced by traversed routers seems still valid, also in the case of one-way measurements. By the same token, we therefore show for the sake of completeness the estimated distance dependency of OWDs in Fig. 26c, which yields to $6.3 - 8.1\mu s/km$ (black solid line/dashed gray line, $7.5\mu s/km$ in Section 4.2.2) and likewise amount to roughly one third above the $5.14\mu s/km$ propagation delays within optical fibers. Anyway, it should be emphasized that this correlation is an average and holds for long-distance (preferably intercontinental) paths only. So, we finally list according to Section 4.2.2:

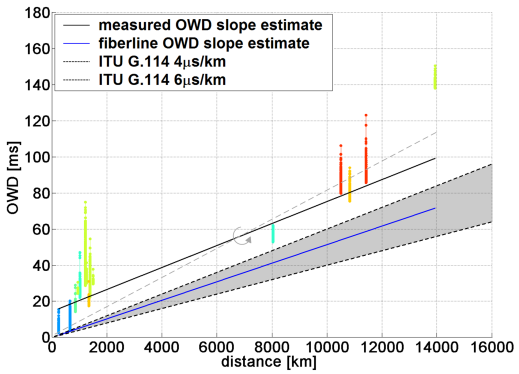
$$OWD_{meas} \approx 1.5 \cdot OWD_{fiber}. \tag{22}$$



(a) Estimated fiberline and measured OWD distributions.



(b) Fiberline OWD vs. measured OWD. Ratio estimate: 1.38 (solid line) – 1.58 (dashed line).



(c) OWD vs. Distance. Slope estimate: $6.3\mu s/km$ (black solid line) – $8.1\mu s/km$ (dashed gray line).

Figure 26: OWD Spatial Correlations.

To further investigate the correlation of OWDs and particular Hop-Counts, we equally took the observed OWDs of remaining ”low-skew” nodes and used robust line fitting over all measured OWD/Hop-Count pairs. From Fig. 27 we therewith notice an increasing trend of OWDs (colored points) with an increasing number of traversed hops, amounting to a slope of $6.3ms/hop$ (solid line). This finding coincides with cognitions from our *traceroute*-based measurements (see Section 4.2.3). Moreover, both measurement campaigns unveiled perfectly comparable results ($6.3ms/hop$ compared to $6.5ms/hop$). Note, that these values still include signal propagation delays in underlying fiber optics. So, to obtain the fraction of pure propagation dependent delays per hop, we further correlated each node's lengthwise OWD (black triangles) with the corresponding Hop-Count and therefrom calculated an estimate of the resulting slope, as shown in Fig. 27 as well (dashed line). From that we recognize a path length dependent delay increase of $4.4ms/hop$, which directly results into an average packet queuing delay of $1.9ms/hop$. Also, this finding perfectly matches with cognitions from our *traceroute* measurements in Section 4.2.3, which revealed a slope of $2.0ms/hop$ in that case.

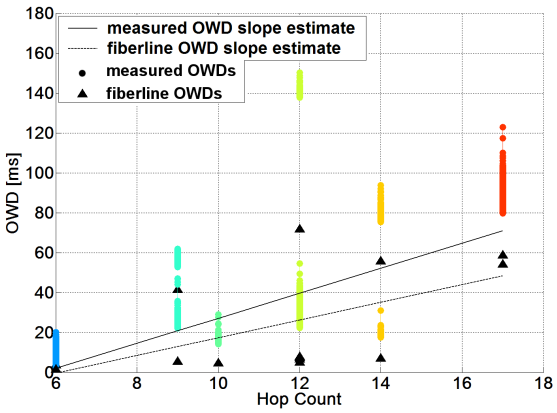


Figure 27: OWD Hop-Count Correlations.

4.3.2 IP Packet Delay Variation

In contrast to OWD, IP Packet Delay Variation (IPDV) is considerably less susceptible to poor synchronization, clock skew and drift. Values between consecutive packets can be taken separately at each node (sender and receiver) and time intervals are usually short, such that the impact of Skew can be neglected.

The IPDVs observed on all paths are drawn in Fig. 28. From that we recognize the majority of values distributed around mean zero and 99% lying within a range below $1ms$. Since IPDVs are calculated for a sequence of packets and two

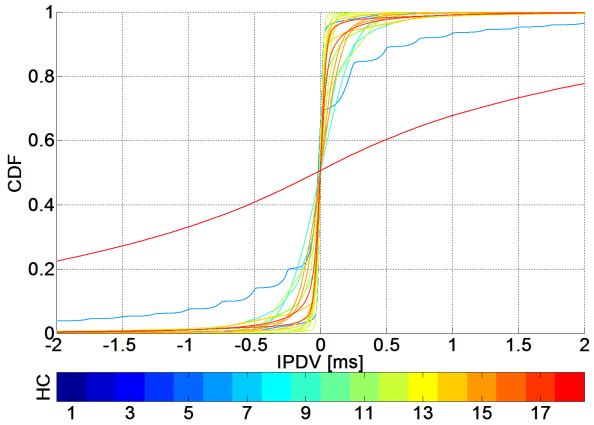


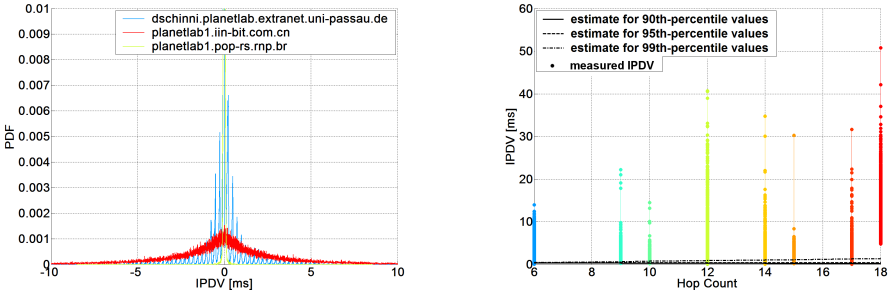
Figure 28: Overall IPDV Statistics.

consecutive packets a time, a more delayed packet (higher delay than prior packet) will cause positive values, while a subsequent packet undergoing again less delay, thus produces negative value. Hence, under normal conditions, i.e. no routers introducing permanently increasing delays etc. are present, we would expect IPDVs exhibiting zero mean. On the evidence of apparent low delay variations we therefore conclude from Fig. 28 that most paths in PlanetLab are either operated within low-capacity regions or traffic volumes are mostly uniform, thus constituting equable router queue utilizations. We further notice, that most observed IPDVs are beyond the determined measurement accuracy (see Section 4.1). Thus, their quantities cannot be treated to be absolute values.

While 18 out of 20 considered paths showed excellent delay variability behaviour, we also observed two routes exhibiting heavy (*planetlab1.iin-bit.com.cn*, (BC)) to medium (*dschinni.planetlab.extranet.uni-passau.de*, (PA)) bias. Fig. 29a depicts the IPDV probability densities of both paths and a low-delay bias node (*planetlab1.pop-rs.rnp.br*, (BR)) for comparison. For BC we obtained IPDV values from $-35.14ms$ to $50.82ms$ with a standard deviation of $3.67ms$ and for PA, IPDVs lie in the range from $-14.52ms$ to $13.99ms$ with a standard deviation of at least $1.11ms$. Since we observed for PA a hop count of only $6hops$, for BR $12hops$ and for BC $18hops$, this already shows, that there might be no significant dependency between IPDV and the number of traversed routers.

As mentioned above, IPDV mean values will mostly tend to zero and thus, delineating them impractical for comparisons of different path IPDVs amongst each other. In order to investigate the correlation of IPDVs and particular hop counts, we rather assess delay variations above a certain probability threshold, i.e. for example all values occuring with a probability of 5% (95th-percentile) or less. Therefore, IPDV values above the 90th, 95th and 99th percentile thresholds of all measured paths are compared against the corresponding Hop-Counts in Fig. 29b. Applying

robust line fitting, we observe a uniform distribution of delay variations over all hops for the above 90% and 95% values, and a slight increase of variability with inreasing Hop-Count for the 99% values. The latter slope however, is caused by the previously mentioned "noisy" path towards BC ($18 hops$). Since most routes revealed evidently low delay variations and slightly higher IPDVs could be observed on only two paths that differ in "length" by more than 12 hops, we could not ascertain an apparent correlation of IPDV and the number of traversed hops.



(a) Top two IPDVs compared to a low IPDV node.

(b) IPDV vs. Hop-Count.

Figure 29: IPDV Hop-Count Correlations.

4.3.3 One-way Packet Loss and Reordering

Extracting sequence numbers carried within the UDP payload of our probe packets and examining sender and receiver PCAP traces allowed us to exactly detect lost packets as well as packets arriving out of order at destined targeted nodes. However, it should be emphasized that observed loss rates and patterns not necessarily reflect the effective conditions on the particular network path under observation, i.e. the fraction of lost packets among *all* transfered traffic and their burstiness, since Poisson probing is used. A detailed discussion on the effects of Poisson probing and loss rate estimation can be found in [64].

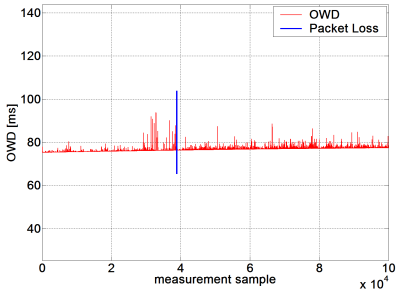
Besides possible inaccuracies, nearly all measured paths showed superior transmission reliability. At 18 out of 20 investigated nodes more than 99.99% of $100,000$ transmitted packets were received without errors. On the remaining two routes on the other hand, 99.971% (*planetlab2.csres.utexas.edu*) and 99.522% (*planetlab1.iin-bit.com.cn*) of packets were transmitted successfully. A listing of all particular loss rates is further given in Table 6. Moreover, all successfully received packets arrived in the same order as they were sent, i.e. no packet reordering was observed at all. Despite some heavier loss rates in the two cases mentioned above, all results still constitute a significant increase of transmission reliabilty compared to measurements by Paxson [47] in 1997 (97.3% and 94.8% successful transmissions).

In order to examine the nature of observed packet loss, whether it occurs in bursts or randomly, as well as possible reasons, we further identified individual time instants of packet loss occurences throughout particular measurements. Since meaningful inferences can only be drawn from observations exhibiting higher loss rates, we only considered *planetlab2.csres.utexas.edu* (University of Texas, Austin, USA, (UOA)) and *planetlab1.iin-bit.com.cn* (Langfang Development Area, Beijing, China, (LDA)) in this step.

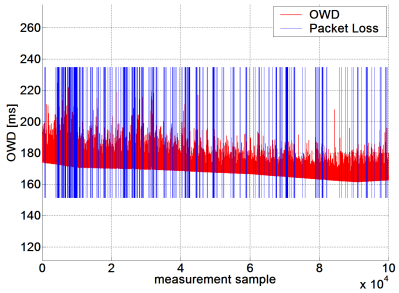
Consequently, Fig. 30a depicts regions of packet loss in connection with measured OWDs over time for UOA. From that we recognize all 29 packet losses occuring within only a short time interval in the first half of the measurement, i.e. in this case packet loss evidently occurs as a singular, bursty event. For further illustration of this bearing, we calculated the UDP-One-Way-Loss-Distance-Stream metric (see Section 3.2.1), unveiling the distance in terms of sequence numbers between lost packets. Fig. 30c therewith affirms that all packets were lost consecutively, since the maximum observed Loss Distance is one. Thus, the corresponding propability density, shown for UOA in Fig. 30e, provides no new information, but is rather given for the sake of completeness. Further on, while all packets were sent successfully throughout the identified loss period, but a preceding increase of OWD could be observed from Fig. 30a, we expect these packets being lost during a high load condition in the network. However, different processes at the receiver side might also be responsible, but could not be determined so far.

From Fig. 30b, likewise showing packet loss and OWD over time for LDA, we observe a completely different behaviour compared to UOA. Losses are rather distributed over the whole measurement window and occur in a random fashion, yet exhibiting some clustering at particular time instants. This bearing is further confirmed by examining the corresponding UDP-One-Way-Loss-Distance-Stream shown in Fig. 30d, indicating individual packet loss occurrences separated by up to $3,500$ sequence numbers. Nonetheless, the majority of loss distances could be observed well below 200, clearly pointing out a bursty loss behaviour. To further examine the nature of such bursts, Fig. 30f depicts the probability densities of Loss Distances. Therefrom we notice 7% of losses accumulated from successive packets (Loss Distance one), still 22% occuring within a 10 sequence numbers range and 51% of losses are less than 100 packets apart. Note, that the obtained loss pattern is "as seen" by the Poisson observer, i.e. packet losses are only sensed in Poisson distributed time intervals and are not monitored in a time continuous fashion. This might lead e.g. to undetected losses during longer time intervals between probe packets. Hence, the real loss pattern might be actually slightly different on the path. However, as in the case of LDA, we further notice loss clustering in areas of high delay variability. Hence, we also expect most packets getting lost during high load conditions in the network. Moreover, considering that the UOA route by far exhibited the highest observed delay variations (see Section 4.3.2), it is not surprisingly to notice the highest packet loss rate in this case as well.

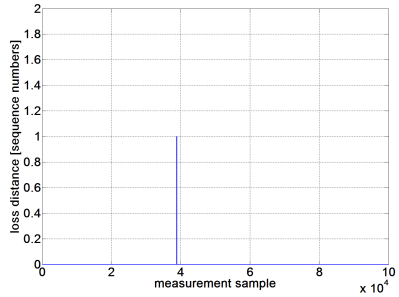
Finally, it is worth noting that packet loss is not directly depending on the number of traversed hops, but is rather coupled with link congestions and failures, which ones on the other hand might occur with inreased probability at likewise inreased Hop-Counts.



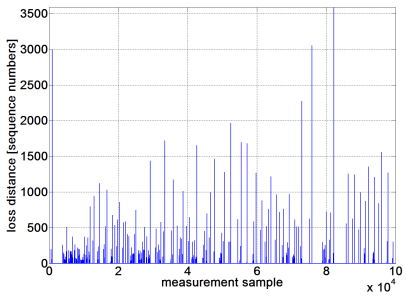
(a) OWD and Packet Loss Occurrences over Measurement Window. University of Texas, Austin.



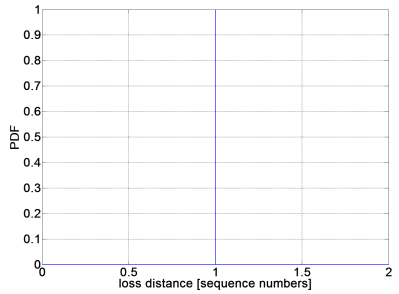
(b) OWD and Packet Loss Occurrences over Measurement Window. Langfang Development Area, Beijing.



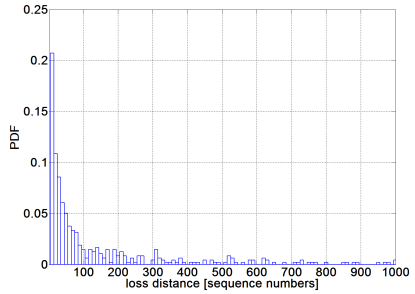
(c) One-way Loss Distance Stream. University of Texas, Austin.



(d) One-way Loss Distance Stream. Langfang Development Area, Beijing.



(e) Loss Distance Distribution. University of Texas, Austin.



(f) Loss Distance Distribution. Langfang Development Area, Beijing.

Figure 30: One-way Packet Loss Statistics.

Target Node	Lost Packets	Loss Rate
dschinni.planetlab.extranet.uni-passau.de	0	0.00
planet2.inf.tu-dresden.de	2	2.00e-5
pl1.bit.uoit.ca	0	0.00
planck227.test.ibbt.be	0	0.00
planetlab2.cs.vu.nl	0	0.00
planetlab1.pop-rs.rnp.br	0	0.00
planetlab2.cs.uoregon.edu	0	0.00
planetlab2.di.unito.it	1	1.00e-5
planetlab2.ifi.uio.no	0	0.00
planetlab2.s3.kth.se	1	1.00e-5
planetlab3.hiit.fi	0	0.00
planetlab4.mini.pw.edu.pl	0	0.00
planet2.scs.stanford.edu	0	0.00
planetlab2.csres.utexas.edu	29	2.90e-4
planetlab3.xeno.cl.cam.ac.uk	0	0.00
planetlab2.eurecom.fr	0	0.00
planetlab2.larc.usp.br	4	4.00e-5
planetlab3.flux.utah.edu	0	0.00
pl-node-0.csl.sri.com	0	0.00
planetlab1.iin-bit.com.cn	478	4.78e-3

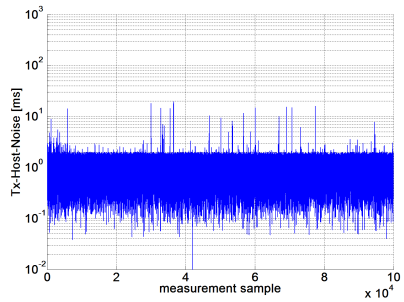
Table 6: One-way Packet Loss Rates.

4.3.4 Tx-Host-Noise

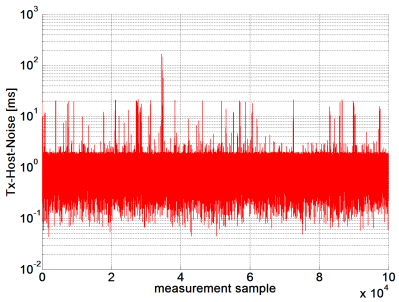
The Tx-Host-Noise metric is a measure that counts for the ”quality” of a sender in terms of its timing precision, i.e. it reflects how accurate pre-calculated inter-packet intervals are adhered at transmission. While this quantity directly determines the accuracy of measured packet timestamps, it is also important in order to assess the impact of timing ”jitter” on the distortion of requested probe stream Poisson distributions. Since the correctness of inferences drawn on the top of the *PASTA* [72] property partially rely on unbiased Poisson processes, results may become unreliable in case of heavy distortions. Certainly, such distortions do not only appear within the sending queue, but rather can occur throughout the entire transmission chain. However, high utilization of sytem resources at the transmitter side may already render the probing stream useless, even before packets are brought on the wire.

The average observed Tx-Host-Noise throughout all measurements amounted to $1.05ms$. Since we already discovered a mean clock resolution of $1ms$ for PlanetLab node operating systems in Section 4.1, the observed noise behaviour is mainly induced by that fact. However, an overview of all obtained Tx-Host-Noise mean

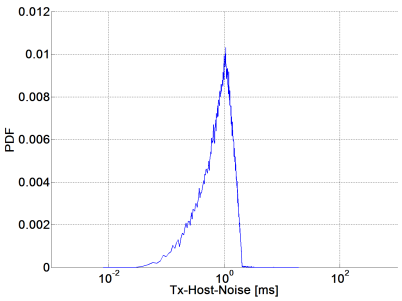
values and standard deviations is listed in Table 7. Further on, we emphasize that, compared to the average value, particular packet queuing delays randomly may become quite higher. To illustrate this behaviour, Tx-Host-Noise values per probe sample of the measurements exhibiting the lowest (*dschinni.planetlab.extranet.uni-passau.de*, Passau) as well as the highest (*planetlab2.cs.vu.nl*, Amsterdam) mean noise are shown in Fig. 31a and Fig. 31b respectively. From that we observe individual packet interval deviations up to $19.5ms$ in the "low-noise" and comparatively worse $168.5ms$ in the "high-noise" case. Further examining the noise distributions of the former ("low-noise") measurement, shown in Fig. 31c, and the latter ("high-noise") measurement, illustrated in Fig. 31d, we recognize equivalent normally distributed shapes in both cases (this even holds for all other cases), except some high-delay outliers within the "high-noise" case. Thereby, the mentioned extreme values only occur to negligible low probabilities, i.e. we observed no more than 0.9% of values exceeding $2.0ms$ Tx-Host-Noise. Further note, that all values always belong to the sending node, i.e. *planetlab01.tkn.tu-berlin.de* in all cases.



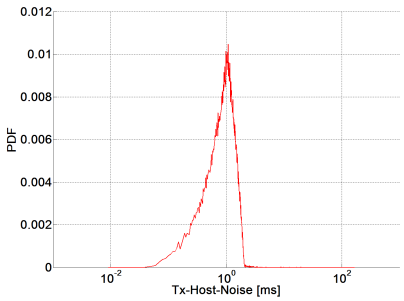
(a) Tx-Host-Noise History. Berlin → Passau.



(b) Tx-Host-Noise History. Berlin → Amsterdam.

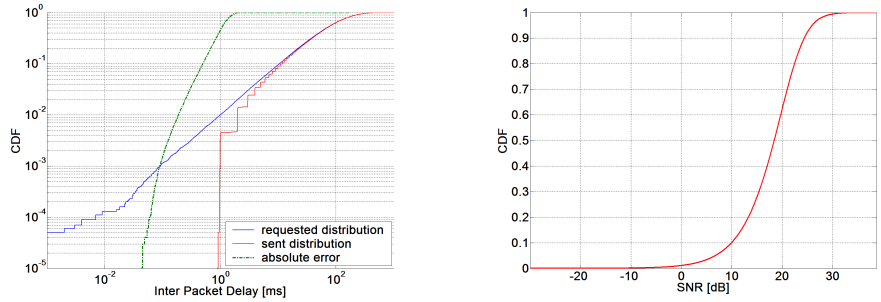


(c) Tx-Host-Noise PDF. Berlin → Passau.



(d) Tx-Host-Noise PDF. Berlin → Amsterdam.

Figure 31: Tx-Host-Noise Comparison.



(a) Comparison of requested and actual Poisson Inter-Packet Intervals. Absolute Error.

(b) Ratio of requested Poisson Inter-Packet Intervals and Tx-Host-Noise. Relative Error (SNR).

Figure 32: Probe Stream Bias due to Tx-Host-Noise.

In order to assess the distortion of requested Poisson-distributed inter-packet spacings, we investigated the amount of transmission intervals between probe samples significantly deviating from the precalculated ones. Since Tx-Host-Noise was normally distributed with mean around $1.05ms$ in all cases, requested inter-packet intervals would be shifted by that amount of time in average. This in fact would not change the shape of the Poisson process, though. More likely, a noticable bias would only occur if the noise variance (or standard deviation respectively) is large compared to particular regions of requested packet transmission intervals. Since the average Poisson transmission rate of $100ms^{-1}$ was comparatively large with respect to Tx-Host-Noise in our case, only a fraction of packets with spacings around $1ms$ would be significantly distorted. Moreover, requested packet intervals clearly below the minimum possible timing resolution of the (kernel) network stack, would not be adhered at all. So, this region of the requested Poisson stream definitely would become biased and the contingent of packets therein substantially impacts the applicability of the *PASTA* property. Fig. 32a, showing both the requested and the actually transmitted probe stream packet intervals for the previously mentioned "high-noise" case, clearly points out a cutoff of possible transmission inter-packet delays at $1ms$. Therefrom, a fraction of 1.0% of all packets were found to be affected by that deficiency. Besides that, the absolute error curve further indicates significant bias up to $2ms$, which corresponds to the observed noise standard deviation of $1.836ms$ in this case. Hence, to assess the overall distortion induced by clock noise, we calculated the (timing precision) *Signal-to-Noise-Ratio* (SNR) for all ($i = 1...n$) probe stream packets, which is the ratio of requested IPDs to Tx-Host-Noise:

$$SNR_i = 10 \cdot \log_{10}\left(\frac{IPD_i}{THN_i}\right)[dB] \quad 1 \leq i \leq n-1. \tag{23}$$

The resulting curve is shown in Fig. 32b. From that we observe 2.1% of transmitted inter-packet intervals differing by up to $3dB$ from the requested intervals,

which we suppose to have a substantial impact on the shape of the Poisson process. While we have not found researches on timing noise and their relation to estimation errors of the *PASTA* property so far, this might require further investigations. Nevertheless, the average SNR, which is the ratio of Poisson process expectation value and noise expectation value ($20dB$ in our case) should be kept as low as possible. In order to improve the SNR, either the average Poisson interval could be increased or the underlying operating system timing noise could be decreased.

Consequently, we expect the choice of the average Poisson transmission interval with respect to the operating system's clock resolution, directly affecting the applicability of the *PASTA* property. The lower the ratio of average Poisson interval and possible clock resolution, the higher the process is distorted due to noise-induced bias. Investigations revealing a quantifiable SNR-dependent *PASTA* estimation error thereby remains future work.

Target Node	**Mean [ms]**	**Std [ms]**
dschinni.planetlab.extranet.uni-passau.de	1.047	0.462
planet2.inf.tu-dresden.de	1.049	0.499
pl1.bit.uoit.ca	1.060	0.467
planck227.test.ibbt.be	1.061	0.536
planetlab2.cs.vu.nl	1.103	1.836
planetlab1.pop-rs.rnp.br	1.063	0.550
planetlab2.cs.uoregon.edu	1.055	0.546
planetlab2.di.unito.it	1.053	0.635
planetlab2.ifi.uio.no	1.047	0.465
planetlab2.s3.kth.se	1.064	0.542
planetlab3.hiit.fi	1.049	0.488
planetlab4.mini.pw.edu.pl	1.047	0.503
planet2.scs.stanford.edu	1.065	0.520
planetlab2.csres.utexas.edu	1.061	0.524
planetlab3.xeno.cl.cam.ac.uk	1.072	0.875
planetlab2.eurecom.fr	1.047	0.484
planetlab2.larc.usp.br	1.047	0.485
planetlab3.flux.utah.edu	1.074	0.629
pl-node-0.csl.sri.com	1.076	0.635
planetlab1.iin-bit.com.cn	1.059	0.474

Table 7: Tx-Host-Noise Statistics.

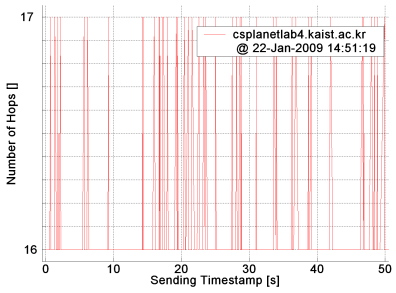
4.3.5 Routing Anomalies

Initially, we considered 22 nodes within our one-way path characteristics measurements. Since the primary scope of this project was to investigate path characteristics with respect to Hop-Counts, we omitted two nodes in the previous analysis due to heavy variations of the IP-header TTL field values. Nonetheless, both measurements revealed unexpected and remarkable behaviour. So, we separately depict the observed routing anomalies in this Section to provide information on that issue, which might give reasons for future investigations. Both measurements were conducted on January 22nd 2009 with particular target nodes and starting times as given in Table 8. Further on, the same probe stream settings as listed in Table 4 also apply here. Noticeably, both target nodes were located in East Asia.

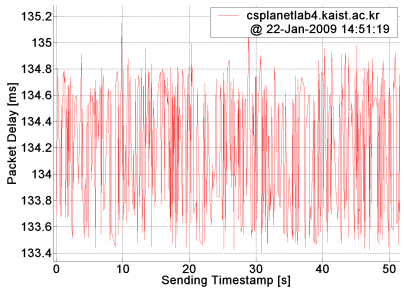
Figure 33 shows measured Hop-Counts and OWDs for the first target node, located at the *Korea Advanced Institute Of Science And Technology*, Seoul. OWD thereby is provided to identify possible route changes, which might be associated with perceptible delay alterations. Obviously, the number of recorded IP hops (TTL decrements) varies between $16hops$ and $17hops$ at a quite high frequency of about 2 alternations per second in average. The observed OWD on the other hand provides no clear evidence for a remarkable route change at particular time instants of Hop-Count modifications. Therefore, we could not definitely gauge the reason for the shown behaviour. Possibly a load-balancing facilty or a false treatment of IP-header TTL fields by responsible routers might be in place.

Target Node	Loc.	Date/Time (UTC)
csplanetlab4.kaist.ac.kr	KR	22.01.2009/14:51:19
planetlab1.iis.sinica.edu.tw	TW	22.01.2009/12:03:19

Table 8: Routing Anomalies. Measurements Schedule.

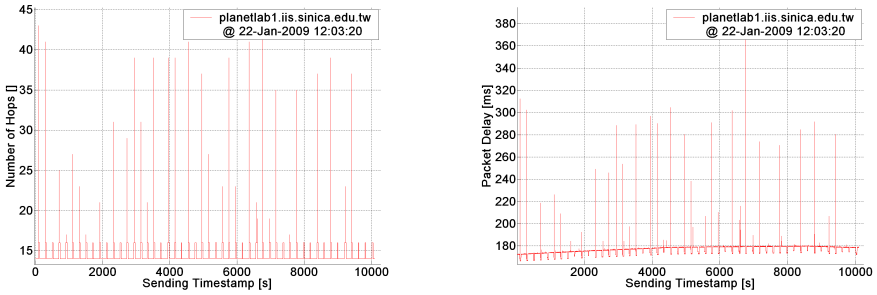


(a) Measured Hop-Count.



(b) Measured OWD (incl. Offset and Skew).

Figure 33: Hop-Count Variations over Time. Seoul, Korea.



(a) Measured Hop-Count.

(b) Measured OWD (incl. Offset and Skew).

Figure 34: Hop-Count Variations over Time. Taipei, Taiwan.

Figure 34, likewise showing Hop-Counts and OWDs for the second target node, located in Taipei, Taiwan, reveals a completely different, but less obscure behaviour. We observed TTL variations between $14hops$ and $44hops$, which on the other hand occur less frequently compared to the previous case. However, the measured OWD evidently changes by roughly $100ms$ in conjunction with each Hop-Count alternation. here. Hence, a significant route change could be a possible reason behind that bearing, but an alternative path differing by $30hops$ in length seems quite unusual.

Finally, resolving the shown routing anomalies and revealing the underlying network structures would require more sophisticated measurement techniques or even better access to the routers themselves. Further investigations on that issues are out of the scope of this report, but might be treated in future work.

4.4 Extended Measurements

Even though path characteristics obtained from synthetic measurements unveil salient network performances, users might experience a subjectively different behaviour. To investigate the relation of observed path characteristics and real-world application quality we conducted several extended measurements in parallel. Above all, our attention was directed to the previously shown striking delay variation performance within PlanetLab (see Section 4.3.2). Thus, we deployed Voice over IP data transfer, which eminently behaves critical in terms of delay and delay variations. Since the latter inherently comes along with network load conditions we also measured Available Bandwidth in order to complete our view.

For the moment our obervations confine to the route Berlin→Berkeley and vice versa only. From Berlin (*planetlab01.tkn.tu-berlin.de*) to Berkeley (*planetlab7.millennium.berkeley.edu*) it takes 17 IP hops as per *traceroute* output and a fairly constant Round Trip Time of $189.5ms$. Along the reverse path on the other hand *traceroute* indicates a slightly higher number of 19 traversed hops, whereas the RTT remains nearly the same. To the best of our knowledge we identified the (forward and reverse) route to be consisting of nine geographically distinct path segments and we further located the routing points between segments as shown in Fig. 35 below. Additionally, the routing points identifiers given in Fig. 35 are listed in Table 9 and exemplary traceroute outputs, targeted from both nodes, are provided in Appendix A.

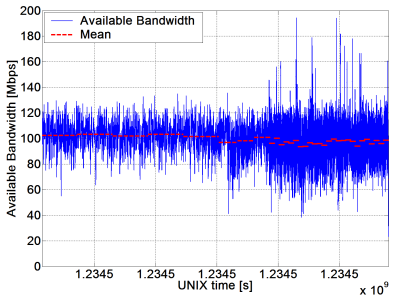
Figure 35: Extended Measurements, Route Berlin↔Berkeley.

Identifier	Location
TUB	Technical University of Berlin, Berlin, Germany
POT	Potsdam, Germany
FRA	Frankfurt (on the Main), Germany
WASH	Washington, USA
ATLA	Atlanta, Georgia, USA
HOUS	Houston, Texas, USA
LOSA	Los Angeles, California, USA
SVL	San Jose, California, USA
OAK	Oakland, California, USA
UCB	University of California, Berkeley, California, USA

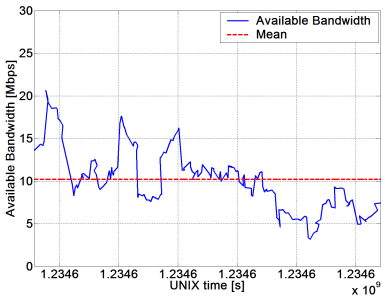
Table 9: Extended Measurements, Routing Points.

4.4.1 Available Bandwidth

To determine the range of bandwidths to deal with, we first investigated the network structure and spotted operators of several path segments, therewith obtaining information on particular configurations. In the order of their appearance from Berlin to Berkeley we identified DFN (X-WiN Backbone) in Germany, GÉANT2 for the transatlantic link, NLR PacketNet for the US east-west traverses and the CENIC education network (CaIREN-HPR Backbone) throughout paths across California. According our inquiries we found that all operators deployed at least 10GE links on the route, while the access network from TUB to DFN as well as the access of UCB to CENIC provide a 1GE connection. Consequently we expected Available Bandwidth values being within the bounds of $1Gbps$.



(a) Available BW, Berlin→Berkeley, 2009-02-13 08:38-20:38 CET. Tight-Link through $100Mbps$ Ethernet Uplink.

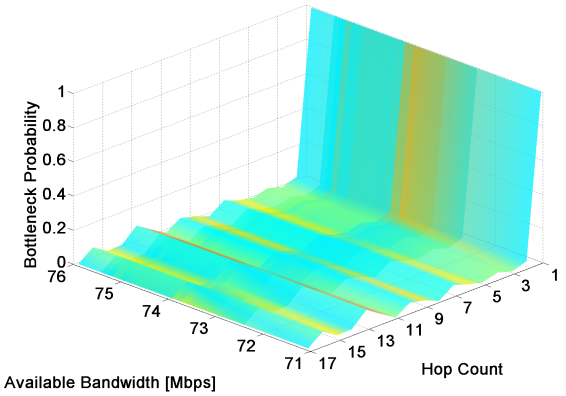


(b) Available BW, Berkeley→Berlin, 2009-02-14 01:28-01:39 CET. Tight-Link through $10Mbps$ PlanetLab BW-Limit.

Figure 36: PathChirp Available Bandwidth Results.

In the first step we used *PathChirp* [54] to observe the Available Bandwidth from Berlin to Berkeley and vice versa. For the former direction we conducted measurements on several days and daytimes which all retained the same result. As shown in Fig. 36a we observed a fairly constant $100Mbps$ Available Bandwidth in average (mean over 1,500 consecutive measurement samples a time, indicated by red lines) over a seven hour period. We noticed a slight increase of the dynamic range within the second half of the measurement but the average Available Bandwidth nearly remains the same. As this finding seems quite unreasonable it rather shows evidence for the presence of a tight link between PlanetLab node and the access router. Owing to the observed bandwidth we presumed a Fast Ethernet connection in this case. In consequence we used *STAB* [53] to locate the tight link and peformed further measurements to verify the previous findings. From that we observed an $80Mbps$ Available Bandwidth and a 100% tight link probability for the first hop. The results for all hops and the corresponding bandwiths are illustrated in Fig. 37. Even though *STAB* reports a slightly smaller bandwidth compared to *PathChirp*, $80Mbps$ are still realistic for Fast Ethernet IP throughput. Unfortunately, an intermediate Ethernet switch was confirmed from our network administration later on, making further measurements in this direction meaningless.

Following we concentrated on the reverse direction from Berkeley to Berlin. While we did not had knowledge of the access network structure in Berkeley, we recognized a $10Mbps$ bandwidth limit from the CoMon database though. However, we conducted *PathChirp* measurements in order to verify the expected $10Mbps$ Available Bandwidth. The obtained results are shown in Fig. 36b and confirm that the PlanetLab bandwidth limitation works properly ($10Mbps$ mean over all measurement samples, indicated by red line). As of the termed restrictions we suspended our effort on Available Bandwidth estimation in PlanetLab for the moment and leave that topic future work.



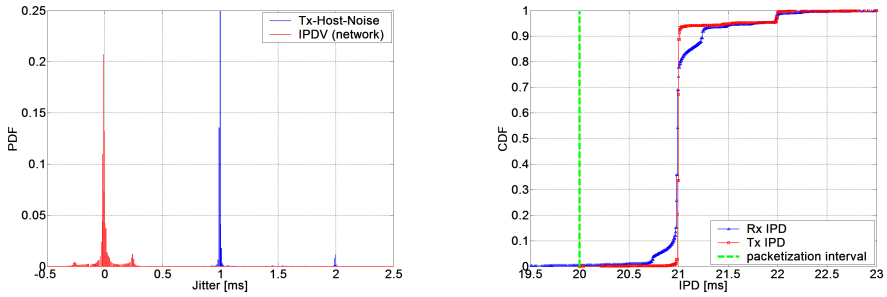
(a) Berlin→Berkeley, 2009-02-15 00:31-05:31 CET.

Figure 37: STAB per Hop Available Bandwidth Results.

4.4.2 Voice over IP

As mentioned earlier in Section 3.1.4 we used our packet-player to transmit pre-recorded VoIP streams in one direction from the imagined caller to the destined callee. We then evaluate path characteristics such as OWD, IPDV, Tx-Host-Noise and Packet-Loss for the particular VoIP call and estimate a voice quality derived from ITU G.114. By utilizing a basic de-jitter buffer model we thus account for the performance of a VoIP client application in order to asses its performance when used over the observed path. In this Section we will present Voice over IP measurement results applied again on the aforementioned route Berlin→Berkeley. We emulated a 1,563 seconds voice call by transmitting 78,175 G.711a coded voice packets via RTP/UDP/IP on Feb 14, 21:19 CET. Therefore each packet contained $160Byte$ voice data resulting in $200Byte$ total IP packet sizes and a $14.91MB$ aggregate transmitted data volume. The effective packetization interval of the vocoder was set to $20ms$.

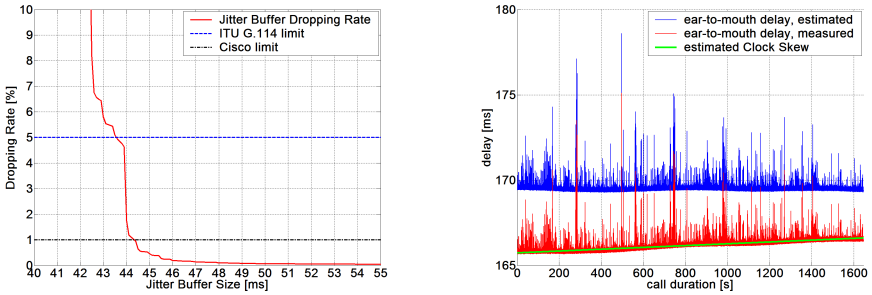
One crucial metric for VoIP quality is the amount of lost packets. From the captured PCAP files we observed 0% Packet Loss as well as no packet was out of order. Since voice streams are constant bit-rate services in most cases their quality is also affected by too excessive delay variations. As shown in Fig. 38a the voice packets experienced quite low delay fluctuations within the network itself. The IPDV exhibits a standard deviation of about only $0.21ms$. This value even outperforms the measured Tx-Host-Noise, exhibiting a standard deviation in the range of $0.33ms$. We also observe the latency of the operating system's process scheduler and network stack in the range of $1ms$ which perfectly coincides with our findings in Section 4.1. We believe this result to be remarkable since it leads to the assumption that the major source of delay variations is not the network itself but rather the network endpoint.



(a) Tx-Host-Noise (mean $1.05ms$, std $0.33ms$); IPDV (mean $0ms$, std $0.21ms$).

(b) IPD. Tx: 0.01% early packets, $32.8\% > 21ms$; Rx: 0.5% early packets, $25.8\% > 21ms$.

Figure 38: VoIP Delay Variation Statistics.



(a) De-Jitter Buffer Dropping Rate. Packet Loss $< 1\%$ at $45ms$ Buffer Size.

(b) Ear-to-Mouth Delay. $170ms$ with Clock Skew and Offset Removal.

Figure 39: VoIP Quality Statistics.

However, from the delay variation's perspective we would not expect much problems with voice quality on the receiver side. Hence, to investigate the inter-packet arrival times on *planetlab7.millennium.berkeley.edu* we determined the corresponding IPDs of each voice packet pair. We observe from Fig. 38b that due to the dominant influence of the process scheduling delays on the sender side nearly all packets arrived out of time, wherby about 26% arrived more than $1ms$ too late.

Since varying packet arrival times are not uncommon and not avoidable at all, de-jitter buffers are fundamental elements of packet voice systems to compensate such variations. But due to voice quality-affecting end-to-end delay constraints their delay compensation ability is limited though. If voice packets arrive out of the "bufferable" time expressed in multiples of the voice codec rate, they cannot be delivered to the decoder and will be dropped. This results in ordinary packet loss as it would occur if packets get lost within the network, thus impairing the experienced voice quality. We used our de-jitter buffer model as defined in Section 3.2.3 and calculated the expected dropping rate dependent on the chosen buffer size in Fig. 39a. From that we obtain at least a required buffer size of $44ms$ to meet the G.114 recommendation or a slightly higher value of about $45ms$ to reach a 1% target dropping rate as proposed e.g. in [63]. Furthermore we also notice a rapid decrease of lost packets with an increasing buffer size which is in coherence with the relatively small delay variation values. Since we did not take into account further "noise" occuring on the receiver side operating system, we would expect the required buffer sizes to be slightly higher.

Last but not least we have to obtain the second quality affecting metric from G.114 yet, ear-to-mouth delay. For this purpose we compute the VoIP delay budget according to Table 3 by adding $10ms$ coder delay, $20ms$ packetization delay and the required $45ms$ de-jitter buffer delay to the measured OWD. As shown in Fig. 39b the Ear-to-mouth delay and therewith the OWD exhibits a continuous

and monotonic increase in time, which indicates relative clock Skew between both nodes to a high probability. By using robust line fitting and on condition that the Skew is linear we obtain a drifting of about $0.559\mu sec/sec$ which we used to correct the OWDs in a further step.

Since both nodes are not highly synchronized through a precise reference clock like GPS we would expect a relative clock Offset between them as well. Hence, in order to estimate the maximum ear-to-mouth delay we used the fairly constantly measured $RTT/2 = 94.5ms$ and therefrom supposed an Offset of about $3.75ms$ by comparing to the mean OWD which amounts to $90.75ms$. However, we remark that the reverse route Berkeley→Berlin at least covers two more hops as per traceroute output, i.e. 19 hops instead of 17 hops are indicated, which turns the above OWD adjustment not necessarily correct. Moreover, due to the lack of relative clock Accuracy a precise Offset determination is not possible at all.

Despite the mentioned uncertainties we would obtain an ear-to-mouth delay of about $170ms$ in the one case or $166ms$ in the other, as can be seen from Fig. 39b, too. This implies only averaging to poor conversational quality according to ITU G.114 on an intercontinental connection. While packet losses owed to delay variations and associated de-jitter buffer droppings are negligible, i.e. only $2 - 3ms$ overall delay could be saved if the IPDV would tend to zero, the quality depreciating part is the OWD.

Thus, a VoIP quality enhancement solution would comprise the reduction of packetization intervals in order to shrink the overall ear-to-mouth delay. The utilization of a vocoder such as the G.726 [66] with a default $10ms$ packetization inerval might be a suitable choice.

5 Concluding Remarks

We used PlanetLab in an attempt to investigate path characteristics dependent on the number of IP hops along several paths in the Internet. Utilizing the *hostip.info* geo-location database as well as PlanetLab's *CoMon* monitoring site in connection with assumptions based on the *ITU G.114* standard, we determined location and distance-related traceroute (round-trip) statistics of about 250 nodes. From these dataset we drafted 20 routes in order to analyze one-way packet dynamics along particular paths between PlanetLab nodes, encompassing a representative range of 6 to 18 IP hops towards selected sites while maintaining a widespread geographic distribution. For this purpose we devised a *PCAP* based measurement framework to be used in PlanetLab, including a packet replay engine which showed to yield comparable performance to *tcpreplay*.

Our measurements unveiled PlaetLab node operating system clock resolutions in the range of $1ms$, which must be treated as a lower bound on time-related measurement accuracy. Moreover, the underlying Linux timer system induced "noise" showed to be normally distributed with mean $1.05ms$ and standard deviations of $0.5 - 1.8ms$, depending on utilization of the particular node. From that, we illustrated that the shape of Poisson sampled probe traffic may becomes considerably biased, if sending rates are chosen too high. Therof, the applicability of the *PASTA* property might become critical. In order to assess the distortion of the Poisson sampling process due to the presence of timer system noise, we introduced a simple "Signal-to-Noise-Ratio" (SNR) metric, quantifying that error. Further investigations on the correlations of such an SNR and the *PASTA* property would help to provide a better understanding of operating system timing uncertainties and their impact on active probing accuracy. Thus, coming along with poor synchronization and partly observed high relative clock skew and drift values, we found PlanetLab not quite applicable for (one-way) packet dynamics measurements yet. Further investigations using hosts outside PlanetLab showed, that simple modifications of the Linux kernel can yield significant timing accuracy improvements, exhibiting clock resolutions in the lower microsecond regime on contemporary x86 hardware. We suppose the observed timing inaccuracies to being mainly a problem of the node's underlying virtualization solutions currently in use. Since we also recognized some inconsistencies in the *hostip.info* database, we would anticipate the enhancement of PlanetLab nodes by the deployment of GPS receivers, therewith gaining a win-win situation and offering new promising research possibilities while providing superior timing and positioning accuracy.

The assessment of delay variability within PlanetLab however, uncovered a striking network performance that even outperformed the packet delay variations induced by the measurement hosts themselves in most cases. We observed the IPDVs of UDP packets to be within the determined measurement accuracy, lying in the range of $1ms$ and below. Expectably, relations of IPDV and particular Hop-Counts could not be ascertained so far. As a consequence thereof additional Voice over IP measurements revealed the overall end-to-end delay to be the lim-

iting factor of voice quality, further noting that packet losses within the network were also quite low in the range of thousandth or non-existent. However, the common assumption of bursty behaviour of packet loss in the Internet could also be proven here in a few cases. Hence, we conclude from our results that besides high-performance networks, proper host setups are alike essential to meet the stringent timing requirements of current and future multimedia services. At the same time we found the *Internet2* prepared to fulfill that requirements, providing sufficient delay variability performance for standard Voice over IP services.

While we could not determine a significant trend in delay variability and the number of traversed hops, we found approximately one third of packet delays induced by router queues and the other two third evoked from propagation delays due to the length of underlying optical fibers. The estimated distance-dependent propagation times thereby were estimated in compliance with values provided by *ITU G.114* and further, a clear dependence of traversed routers and a constant increase in overall packet delay could be observed. According our estimates we quantify the constant delay increase per hop by about $2ms/hop$ in average, emphasizing that we expect this value being valid for a large-scale (intercontinental) environment only. The mentioned dependencies therby were independently derived from round-trip (*traceroute*) and one-way measurements results as well. Since we identified end-to-end delays to be the limiting factor in voice quality, a reduction of the number of traversed routers would help to improve packet voice performance on at least intercontinental WAN connections.

Our analysis of hop counts during one-way packet dynamics measurements also disclosed some obscure behaviour we were not able to clearly bear upon. In the one case we recorded alternating TTL entries of $16 - 17hops$, occuring at least 5 times every $10s$ throughout the whole measurement duration of $168min$. This rapid hop count variation may be evoked by the presence of a load-balancing facility, but the permanence and frequency of changes seem quite uncommon for such an assumption. In the other case we observed a less frequent but rather high Hop-Count variation between $14 - 44hops$ on the other hand. We would not expect load-balanced paths differing by more than 20-30 hops in length, delineating that finding quite implausible. However, a false treatment of IP header TTL fields by the correspondent routers might be a possible explanation, but further investigations are needed to unfold the particular network topology and resolving that behaviour. The mentioned contradictions point out once more the need for sophisticated topology discovery mechanisms beyond traceroute.

References

[1] G. Almes, S. Kalidindi, and M. Zekauskas. RFC 2679 (rfc2679) - A One-way Delay Metric for IPPM, September 1999.

[2] G. Almes, S. Kalidindi, and M. Zekauskas. RFC 2680 (rfc2680) - A One-way Packet Loss Metric for IPPM, September 1999.

[3] G. Almes, S. Kalidindi, and M. Zekauskas. RFC 2681 (rfc2681) - A Round-trip Delay Metric for IPPM, September 1999.

[4] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 153–158, 2006.

[5] F. Baccelli, S. Machiraju, D. Veitch, and J.C. Bolot. The Role of PASTA in Network Measurement. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 231–242, New York, NY, USA, September 2006. ACM.

[6] D. Bickson. Safe Planetlab Raw Sockets. DANSS LAB (Distributed Algorithms Networking and Secure Systems Group), The Hebrew University of Jerusalem, http://www.cs.huji.ac.il/labs/danss, June 2004.

[7] J.C. Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In *Proc. SIGCOMM '93*, pages 289–298, September 1993.

[8] D.P. Bovet and M. Cesati. *Understanding the Linux Kernel*. O'Reilly, 3rd edition, November 2005.

[9] R.L. Carter and M.E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. Tech. Report BU-CS-96-006, Computer Science Department, Boston University, March 1996.

[10] R.G. Cole and J.H. Rosenbluth. Voice over IP Performance Monitoring. *SIGCOMM Comput. Commun. Rev.*, 31(2):9–24, 2001.

[11] C. Demichelis and P. Chimento. RFC 3393 (rfc3393) - IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), November 2002.

[12] DFN-Verein e. V.. Deutsches Forschungsnetz (DFN). http://www.dfn.de, March 2009.

[13] A. Turner et al. Tcpreplay, Pcap editing & replay tools for *NIX. http://tcpreplay.synfin.net, January 2009.

[14] T. Gleixner and D. Niehaus. Hrtimers and Beyond: Transforming the Linux Time Subsystems. In *Proceedings of the Linux Symposium*, volume 1, pages 333–346, July 2006.

[15] R. Govindan and H. Tangmunarunkit. Heuristics for Internet Map Discovery. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1371–1380, 2000.

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

[17] M. Handley, V. Jacobson, and C. Perkins. RFC 4566 (rfc4566) - SDP: Session Description Protocol, July 2006.

[18] K. Harfoush, A. Bestavros, and J. Byers. Measuring Bottleneck Bandwidth of Targeted Path Segments. In *INFOCOM 2003*, volume 3(30), pages 2079–2089, April

2003.

[19] S. Hemminger. Network Emulation with NetEm. In *Proceedings of the 6th Australia's National Linux Conference (LCA2005)*, April 2005.

[20] hostip.info. IP Address Lookup and GeoTargeting Community Geotarget IP Project. http://www.hostip.info, January 2009.

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

[22] Cisco Systems Inc. Understanding Delay in Packet Voice Networks - Document ID: 5125. http://www.cisco.com, February 2006.

[23] VMware Inc. Timekeeping in VMware Virtual Machines. http://www.vmware.com, August 2008.

[24] University of Southern California Information Sciences Institute. RFC791, INTERNET PROTOCOL - DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, September 1981.

[25] University of Southern California Information Sciences Institute. RFC793, TRANSMISSION CONTROL PROTOCOL - DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, September 1981.

[26] European Telecommunications Standards Institute. Network Aspects (NA); Availability performance of path elements of international digital paths. Rec. EN 300 416, ETSI, August 1998.

[27] V. Jacobson. traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.gz, February 1989.

[28] M. Jain and C. Dovrolis. End-to-End Available Bandwidth: Measurement methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Transactions on Networking*, 11(4):pp. 537–549, August 2003.

[29] M. Jain and C. Dovrolis. Ten fallacies and pitfalls on end-to-end available bandwidth estimation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 272–277, New York, NY, USA, 2004. ACM.

[30] R. Kapoor, L.J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi. CapProbe: a simple and accurate capacity estimation technique. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2004. ACM.

[31] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Proc. SIGCOMM '91*, pages 3–15, September 1991.

[32] R. Koodli and R. Ravikanth. RFC 3357 (rfc3357) - One-way Loss Pattern Sample Metrics, August 2002.

[33] K. Lai and M. Baker. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay. In *Proceedings of ACM SIGCOMM*, pages 283–294, 2000.

[34] H.J. Leen, M.S. Kim, J.W. Hong, and G.H. Lee. QoS parameters to network performance metrics mapping for SLA monitoring. KNOM Rev. 5 (2), 2002.

[35] M.A. Lombardi, L.M. Nelson, A.N. Novick, and V.S. Zhang. Time and Frequency Measurements Using the Global Positioning System. In *Cal Lab Int. Jour. of Metrology*, pages 26–33. National Institute of Standards and Technology Time and Frequency Division, July–September 2001.

[36] Z.M. Mao, J. Rexford, J. Wang, and R.H. Katz. Towards an accurate AS-level traceroute tool. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications,*

technologies, architectures, and protocols for computer communications, pages 365–378, 2003.

[37] H. Marouani and M.R. Dagenais. Internal Clock Drift Estimation in Computer Clusters. *J. Comp. Sys., Netw., and Comm.*, 2008(2):1–7, 2008.

[38] B. Mel, M. Bjrkman, and P. Gunningberg. Regression-Based Available Bandwidth Measurements. In *International Symposium on Performance Evaluation of Computer and Telecommunications Systems*, 2002.

[39] D.L. Mills. RFC 1305 (rfc1305) - Network Time Protocol (Version 3) Specification, Implementation and Analysis, March 1992.

[40] S. B. Moon, P. Skelly, and D. Towsley. Estimation and Removal of Clock Skew from Network Delay Measurements. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 227–234, March 1999.

[41] H. Moritz. Geodetic Reference System 1980. *Journal of Geodesy*, 74(1):128–162, 2000.

[42] Internet2 Board of Trustees. Internet2. http://www.internet2.edu, March 2009.

[43] K. Park and V.S. Pai. CoMon: A Mostly-Scalable Monitoring System for PlanetLab. *SIGOPS Oper. Syst. Rev.*, 40(1):65–74, 2006.

[44] K. Park and V.S. Pai. CoMon - A Monitoring Infrastructure for PlanetLab. http://comon.cs.princeton.edu, March 2009.

[45] V. Paxson. Towards a Framework for Defining Internet Performance Metrics. In *Proc. INET '96*, June 1996.

[46] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(2):pp. 601–615, October 1997.

[47] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, Computer Science Division - University of California, Berkeley, April 1997.

[48] V. Paxson. On Calibrating Measurements of Packet Transit Times. In *SIGMETRICS Perform. Eval. Rev.*, volume 26(1), pages 11–21, June 1998.

[49] V. Paxson. End-to-End Internet Packet Dynamics. *IEEE/ACM Transactions on Networking*, 7(3):pp. 277–292, June 1999.

[50] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. RFC 2330 (rfc2330) - Framework for IP Performance Metrics, 1998.

[51] J. Postel. RFC 792, INTERNET CONTROL MESSAGE PROTOCOL - DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, September 1981.

[52] D. Kaminsky (DoxPara Research). paratrace, Parasitic Traceroute via Established TCP Flows & IPID Hopcount. http://www.doxpara.com, January 2009.

[53] V.J. Ribeiro, R. Riedi, and R. G. Baraniuk. Spatio-Temporal Available Bandwidth Estimation for High-Speed Networks. In *ISMA 2003 Bandwidth Estimation Workshop (Best)*, December 2003.

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

[55] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta. RFC 3032 (rfc3032) - MPLS Label Stack Encoding, January 2001.

[56] E. Rosen, A. Viswanathan, and R. Callon. RFC 3031 (rfc3031) - Multiprotocol Label Switching Architecture, January 2001.

[57] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC3261 (rfc3261) - SIP: Session Initiation Protocol, June 2002.

[58] S. Rostedt and D.V. Hart. Internals of the RT Patch. In *Proceedings of the Linux Symposium*, volume 2, pages 161–172, June 2007.

[59] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC3550 (rfc3550) - RTP: A Transport Protocol for Real-Time Applications, July 2003.

[60] R. Sherwood, A. Bender, and N. Spring. DisCarte: A Disjunctive Internet Cartographer. *SIGCOMM Comput. Commun. Rev.*, 38(4):303–314, 2008.

[61] R. Sherwood and N. Spring. Touring the Internet in a TCP Sidecar. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 339–344, 2006.

[62] R. W. Sinnott. Virtues of haversine. *Sky and Telescope*, 68(2):158, 1984.

[63] T. Szigeti and C. Hattingh. *Quality of Service Design Overview*. Cisco Press, December 2004.

[64] M.M.B. Tariq, A. Dhamdhere, C. Dovrolis, and M. Ammar. Poisson versus periodic path probing (or, does PASTA matter?). In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 10–10, 2005.

[65] tcpdump.org. PCAP - Packet Capture library. http://www.tcpdump.org, March 2009.

[66] THE INTERNATIONAL TELEGRAPH and TELEPHONE CONSULTATIVE COMMITTEE (CCITT). 40, 32, 24, 16 kbit/s ADAPTIVE DIFFERENTIAL PULSE CODE MODULATION (ADPCM). Recommendation G.726, INTERNATIONAL TELECOMMUNICATION UNION, 1990.

[67] INTERNATIONAL TELECOMMUNICATION UNION. Pulse code modulation (PCM) of voice frequencies. Recommendation G.711, ITU-T, Novemver 1988.

[68] INTERNATIONAL TELECOMMUNICATION UNION. Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. Recommendation P.862, ITU-T, March 2001.

[69] INTERNATIONAL TELECOMMUNICATION UNION. One-way transmission time. Recommendation G.114, ITU-T, May 2003.

[70] S. Walberg. Linux Journal HOWTOs - Expose VoIP Problems Using Wireshark. http://www.linuxjournal.com, March 2007.

[71] J. Wang, J. Yang, H. Zhou, G. Xie, and M. Zhou. Measuring One-way Delay with Multiple Clock Dynamics. In *PDCAT'2003. Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 424–429, August 2003.

[72] R.W. Wolff. Poisson Arrivals See Time Averages. *Operations Research*, 30(2):pp. 223–231, 1982.

Appendices

A Extended Measurements Route

traceroute Berlin→Berkeley:

```
traceroute to planetlab7.millennium.berkeley.edu (169.229.50.9), 30 hops max, 40 byte packets
 1  130.149.49.1 (130.149.49.1)  0.483 ms  0.380 ms  0.610 ms
 2  xr-tub1-ge8-3.x-win.dfn.de (188.1.33.81)  0.598 ms  0.588 ms  0.886 ms
 3  zr-pot1-te0-7-0-3.x-win.dfn.de (188.1.144.221)  2.140 ms  2.129 ms  2.119 ms
 4  zr-fra1-te0-7-0-0.x-win.dfn.de (188.1.145.205)  15.716 ms  15.710 ms  15.698 ms
 5  dfn.rt1.fra.de.geant2.net (62.40.124.33)  15.124 ms  15.123 ms  15.112 ms
 6  abilene-wash-gw.rt1.fra.de.geant2.net (62.40.125.18)  108.031 ms  108.321 ms  107.984 ms
 7  so-0-0-0.rtr.atla.net.internet2.edu (64.57.28.6)  146.122 ms  155.126 ms  155.128 ms
 8  so-3-2-0.rtr.hous.net.internet2.edu (64.57.28.43)  146.447 ms  146.376 ms  146.367 ms
 9  so-3-0-0.rtr.losa.net.internet2.edu (64.57.28.44)  177.000 ms  177.037 ms  177.017 ms
10  hpr-lax-hpr--i2-newnet.cenic.net (137.164.26.132)  179.531 ms *  179.465 ms
11  svl-hpr--lax-hpr-10ge.cenic.net (137.164.25.13)  188.171 ms  187.698 ms *
12  oak-hpr--svl-hpr-10ge.cenic.net (137.164.25.9)  188.865 ms  188.751 ms *
13  hpr-ucb-ge--oak-hpr.cenic.net (137.164.27.130)  189.155 ms  189.151 ms  189.163 ms
14  t2-3.inr-201-eva.Berkeley.EDU (128.32.0.37)  189.152 ms  189.339 ms  189.317 ms
15  evansBB-ptp-inr201.Millennium.Berkeley.EDU (169.229.51.225)  189.307 ms  189.292 ms  189.228 ms
16  sodaBB-ptp-evansBB.Millennium.Berkeley.EDU (169.229.51.185)  189.457 ms  189.457 ms  189.447 ms
17  planetlab7.Millennium.Berkeley.EDU (169.229.50.9)  189.437 ms  189.428 ms  189.418 ms
```

traceroute Berkeley→Berlin:

```
traceroute to planetlab01.tkn.tu-berlin.de (130.149.49.136), 30 hops max, 40 byte packets
 1  planetlab-gw.Millennium.Berkeley.EDU (169.229.50.1)  0.463 ms  0.434 ms  0.412 ms
 2  169.229.51.197 (169.229.51.197)  0.370 ms  0.353 ms  0.333 ms
 3  g3-7.inr-202-reccev.Berkeley.EDU (169.229.51.230)  156.176 ms  156.155 ms  156.357 ms
 4  xe-1-0-0.inr-002-reccev.Berkeley.EDU (128.32.0.38)  1.080 ms  1.059 ms  1.037 ms
 5  hpr-oak-hpr--ucb-ge.cenic.net (137.164.27.129)  1.742 ms * *
 6  svl-hpr--oak-hpr-10ge.cenic.net (137.164.25.8)  3.154 ms  2.775 ms *
 7  lax-hpr--svl-hpr-10ge.cenic.net (137.164.25.12)  10.635 ms * *
 8  nlr-packetnet--hpr-lax-hpr.cenic.net (137.164.26.131)  11.530 ms  11.449 ms  11.427 ms
 9  hous-losa-87.layer3.nlr.net (216.24.186.31)  42.612 ms  42.622 ms  42.604 ms
10  atla-hous-70.layer3.nlr.net (216.24.186.9)  66.999 ms  66.354 ms  66.139 ms
11  wash-atla-64.layer3.nlr.net (216.24.186.21)  81.171 ms  81.339 ms  80.816 ms
12  newy-wash-98.layer3.nlr.net (216.24.186.22)  86.512 ms  86.537 ms  86.370 ms
13  216.24.184.86 (216.24.184.86)  169.527 ms  169.339 ms  169.409 ms
14  so-6-2-0.rt1.fra.de.geant2.net (62.40.112.57)  174.535 ms  174.775 ms  174.731 ms
15  dfn-gw.rt1.fra.de.geant2.net (62.40.124.34)  175.165 ms  175.223 ms  175.204 ms
16  zr-pot1-te0-7-0-2.x-win.dfn.de (188.1.145.138)  188.283 ms  188.261 ms  188.670 ms
17  xr-tub1-te2-3.x-win.dfn.de (188.1.144.222)  189.338 ms  189.293 ms  189.345 ms
18  kr-tu-berlin.x-win.dfn.de (188.1.33.82)  189.357 ms  189.766 ms  189.801 ms
19  planetlab01.tkn.TU-Berlin.DE (130.149.49.136)  189.496 ms  189.474 ms  189.476 ms
```