High Precision Active Probing for Internet Measurement

Attila Pásztor <Attila.Pasztor@eth.ericsson.se>
Ericsson Hungary R&D / EMULab at the dept. of E&EE, The University of Melbourne
Hungary

Darryl Veitch <D.Veitch@ee.mu.oz.au>
EMULab at the dept. of E&EE, The University of Melbourne
Australia

Abstract

Internet measurement based on active measurement techniques is becoming a fundamental part of performance management systems. The increasing importance of these methods is due to their fundamentally end-to-end nature and great flexibility. However, in order to remain non-invasive, active measurement methods in wide use today are typically restricted to measurements on coarse time scales of seconds or minutes. The goal of our work is to create and explore the benefits of a highly accurate active probing system for very fine time scale measurements. A structured overview of network-edge measuring approaches, and issues in accurate traffic monitoring and non-invasive active probing, will be given. Traffic metrics that can be derived from available measurements will also be introduced. A wide range of measurement infrastructures will be presented and compared, including highly accurate systems with GPS synchronisation and special measurement cards, and systems based on off the shelf PC's. Finally, to demonstrate the benefits of high precision active probing, some measurement results will be presented and possible applications outlined.

Contents

1. Introduction

Measurement techniques are traditionally used in telecommunications networks to support a wide range of activities including network planning and design, network operation and research .

A classic example of a traditional technique to obtain measurements would be the various traffic counters built into network switching equipment. Passive measurement of this type is in principle ideally suited for monitoring such quantities as backbone link utilisation, or understanding and modeling aggregate link traffic for dimensioning purposes, and has dominated tele-traffic studies in the past. Obtaining end-to-end measures from collections of such data however constitutes a difficult, data intensive, and time consuming inversion task, motivating the direct measurement of end-to-end network performance metrics, as well as of quality of service measures.

In contrast to traditional measurement, we consider measurements made at the network edge, and focus not on aggregate statistics, but on individual data flows between specific origins and/or destinations. Note that the concepts of "core" and "edge" remain relative. The origins and destinations on the "edge" of the network could mean end users, or alternatively nodes at the edge of some core network..

An important reason for moving away from traditional core-network measurements is that performing them on a large scale, routine basis, implies building measurement infrastructure into the network elements, such as routers, themselves. As traffic measurement will always be a secondary consideration for router manufacturers, obtaining measurements which are adequate both in terms of their time resolution and descriptive ability, is unlikely. This is particularly true during high load where measurements are the most relevant, but the routers the most preoccupied. Furthermore the kinds of measurements required changes over time, and it is difficult to build a sufficiently generic set into a switch which is also high performance. We therefore focus on newer network-edge based methodologies, based on monitoring packet streams in fine to very fine detail, rather than examining counters or control information from network elements.

Active measurement, where controlled probe traffic is generated, injected into a network, and measured at a receiving

node, is becoming increasingly important due its great flexibility, intrinsically end-to-end nature, and freedom from the need to access core network switching elements. Existing large scale active measurement programs [1], [2], [3], [4], [5] have used probe traffic to measure connectivity, delay, and loss statistics on coarse time scales, seconds or more commonly, minutes. Typically simple probe streams have been used such as isolated probes, and low rate periodic or (pseudo) Poisson streams. On finer time scales, several milliseconds to seconds, periodic as well as more sophisticated probe streams have been employed to measure bottleneck bandwidth [6], [7], [8], [9], [10] and available bandwidth [7], [10], and to investigate the detailed statistical structure of delay and loss [7], [11], [12], [13], [14]. However, at such time scales timing problems in common measurement infrastructures can result in measurement errors of the order of the time intervals one is trying to measure, and the inter-departure times of the packet probes one is trying to control. Post processing techniques aimed at redressing or at least monitoring these errors [11], [15], can only imperfectly address a subset of the issues involved. Even for the less demanding low resolution measurements, errors in individual measurements of delay are no less significant, even if less critical.

The primary goal of this paper is to report on active probing systems which are both highly accurate and which allows high resolution measurements and probe stream definition. In addition to enabling the detailed modelling of the structure of losses and delays suffered by probe streams, high accuracy increases the reliability and power of tools for the measurement of network metrics such as bottleneck bandwidth, and makes feasible the broader exploration of the scarcely tapped potential of probing methods.

2. Internet Measurement

Measurements collected in the Internet, organised into the functional groupings as described in [29], are those focusing on the areas of topology (mapping, connectivity), workloads, performance, and routing. Topological data describes the network link infrastructure on different protocol layers. Workload measurements are those related to the resource usage of routers or switches and the utilisation of links. Performance measurements focus on the analysis of end-to-end behaviour and on the diagnosis of network problems. Routing measurements provide insights into the dynamics of routing protocols such as routing table updates. These are of great importance as the reliability and robustness of the Internet depends on the stability and efficiency of routing.

Passive and active measurement are the two fundamental approaches used in communication networks. By passive measurement we mean the standard approach of tracking the performance and behaviour of packet streams simply by monitoring the traffic passing by the measurement point(s). By active measurement we mean the injection of artificial probe traffic into the network, and the measurement of its characteristics at different points, typically back at the origin (round-trip end-to-end measurement), or at some terminating destination (one-way end-to-end).

Passive measurements are usually used to measure metrics pertaining to a certain network element, that is at-a-point metrics such as link throughput and packet size statistics. However from the application point of view, particularly real-time applications, end-to-end quality of service metrics are primordial, and for these the passive approach is inappropriate as the presence of traffic between the end points is not guaranteed. Thus active measurement methods are typically used to obtain end-to-end statistics such as latency, loss and route availability.

While workload and routing measurements typically utilise passive measurements, performance and topology measurements rely on active measurement methods to a large extent. In spite of this general classification the choice of the most appropriate measurement technique to measure a certain metric will depend on the actual circumstances and requirements, and in most cases both passive and active measurement techniques can be applied. To make the right choice a good understanding of the advantages and disadvantages of these fundamental approaches is indispensable.

3. Active Probing

Aside from being able to generate traffic at will between selected points, there are many other advantages of active measurement. Chief among these is the flexibility to design probe streams with particular properties to match measurement requirements: if we think of the route followed by the probes as a black box, then by sending in different test input `signals' and measuring the transformed output, we can shift the measurement focus from simple quantities such as average delay and loss on a route, to bottleneck and available bandwidth, and even to cross traffic estimation [6], [18].The design parameters are the sequence of packet types, sizes and inter-departure times, and can be chosen according to any deterministic and/or statistical criteria. Other advantages include enormously reduced volume of measurement data compared to the passive monitoring of high bandwidth links, and the avoidance of data privacy issues and the corresponding need for payload anonymization.

The main disadvantage of active measurement is its invasive character. The probes modify the very route conditions, and perturb the very traffic, that one is trying to measure. In an attempt to minimize these effects measurement projects typically use probe streams of average bandwidth lower than 10kbps [19]. Since these streams are periodic or pseudo Poisson (the two are very similar in practice), such low rate streams correspond to sampling at low time resolution. The challenge for probe design is to deepen the time resolution without unduly increasing the average bit rate.

3.1 Timing Control

Obtaining accurate timing information is essential to minimize the measurement errors both at the monitors and at the sender. The sender needs to send probe packets at the designated target times, while the monitors must generate accurate time stamps for packet arrivals.

The highest accuracy available is that of dedicated hardware for probe stream generation and monitoring. We use the purpose built DAG3.2e measurement cards for monitoring as described below, but find that RT-Linux offers a software solution for the Sender with adequate accuracy. While this monitoring solution is expensive, other aspects of the system are not. In common with the majority of measurement projects in the Internet community, we focus on commercial PCs running Unix operating systems such as FreeBSD and Linux. RT-Linux, although less common and less mature, is also inexpensive.

We use the hardware monitors not only in the experimental infrastructure itself, but as a reliable reference system with which to investigate the accuracy of Unix based software solutions in general, and the RT-Linux Sender in particular.

To discuss issues related to clock accuracy in more detail some basic terminology is needed. The clock resolution is the smallest unit by which a clock's time is updated. The offset is the difference between the time reported by the clock and the true reference time at a particular moment . The skew is roughly the difference between the clock's frequency and a reference frequency, or more precisely the coefficient of the linear component of the offset.

Our clock skew and stability measurements performed on commercial PCs running Linux are consistent with the results of [20] stating that the clock skew of commercial PCs is typically in the 50 PPM range, with stability of the order of 0.1 PPM. These values can be used to estimate the range of measurement errors of clocks not synchronized to a reference source.

Unfortunately the fine clock resolution available on some processors does not imply low skew, nor high oscillator stability, nor accurate initialization to absolute time. At boot time the software clock is initialized either from the hardware clock or from an external reference. However, in addition to the skew the frequency of the quartz oscillator varies over time due to a number of different factors including changes in the ambient temperature, power level and ageing. Thus besides accurate initialization the clock also has to be continuously corrected for the skew and the smaller time scale effects of oscillator stability [22].

The Network Time Protocol (NTP) is used by the Internet to synchronize computer clocks to an external reference source. The servers providing the synchronization service are hierarchically organized, with primary servers synchronized directly to external reference clocks such as GPS, and secondary servers synchronized to primary servers and others in the synchronization subnet. The clock synchronization is based on the exchange of time stamps between the client and the server, and possibly some other peer. The round-trip delay and the clock offset is then estimated from these time stamps. An upper bound in the offset error estimation is half of the round-trip time [23] , which is in the order of milliseconds even in the case of LANs.

Errors of the order of milliseconds, such as those generated by NTP based synchronization, primarily affect the precision of one-way metrics such as one-way delay, where the accurate synchronization of the monitoring host clocks is essential. However a range of other performance metrics, such as round-trip time, inter-arrival time and delay variation, do not require absolutely synchronized clocks. The known stability of system clocks can then result in much higher accuracy then what is achievable using NTP. This implies that for the measurement of metrics involving small time scales using NTP is not only unnecessary, it could even significantly degrade the measurement accuracy.

In cases when the accuracy of network based clock synchronization is not sufficient, GPS receivers can be used instead as high precision external references. The accuracy of modern GPS receivers output signal is well below 1ms. Large scale measurement projects collecting statistics of one-way metrics such as Surveyor [2], RIPE [3] and AMP [24], all use GPS synchronized measurement hosts. For monitoring high bandwidth links an accuracy of 10ms is not sufficient. In these cases special measurement cards like the DAG series, capable of synchronizing to a GPS signal with much greater accuracy, can be used.

The DAG series cards (see http://dag.cs.waikato.ac.nz/) are special hardware boards designed primarily for the purpose of high accuracy and high performance passive monitoring. They incorporate their own processor and PCI interface. The main design aims of the DAG series includes high performance capture up to OC48 and beyond, on board intelligence to filter data before it is passed to the host processor, industry standard interfaces, programmable and re-configurable structure, and highly accurate timing referenced to a universal time standard [16]. We used two DAG3.2e cards designed for monitoring 10/100 Mbit/s Ethernet, synchronized to a GPS receiver, yielding a time stamping accuracy below 100ns. They were used not only as the monitoring components of the probing infrastructure but as timing references in all of the comparison experiments. The hardware design of these cards allows them to be used as traffic generators, and to perform filtering. However the firmware does not currently support either of these.

The periodic timer interrupts used to increment the system clock counter in Linux and BSD form the basis of the allowed times for the scheduling of all tasks for execution, both system and user. This finite scheduling granularity can not be improved as simply as in the case of the clock resolution, where a processor register counting CPU clock cycles can be used for interpolation at intermediate time points. The default value in both operating systems is 100, corresponding to 100 interrupts per second resulting in 10 ms scheduling resolution. Unfortunately finite resolution is not the only problem. All the periodically executed tasks and timed events queue behind these interrupts, leading to synchronization, increased delays, and timing errors.

The sending of a precise probe stream is a real-time task which is ill-suited to the imposition of a rigid scheduling grid. It is therefore natural to include a real-time operating system in our comparison.

Real-Time Linux [17] was selected because it is implemented as an extension to the Linux operating system. It provides

the capability of running real-time tasks on the same machine as Linux, acting as a lower layer which Linux sees as part of the hardware. The RT-Linux tasks execute whenever they need to, independently of the status of the tasks under Linux. The performance of a real-time task is determined mainly by hardware characteristics such as interrupt latency. Process pre-emption is now strictly limited, being possible only through hardware interrupts or possibly other RT-Linux tasks as determined by the RT scheduler. In this paper only one RT task was ever running, eliminating this latter possibility.

Another extremely useful feature is that the real-time tasks can be connected to ordinary Linux processes, either via a device interface or through shared memory. As a result only real-time critical parts have to be implemented in RT-Linux, other parts of the application can be run as normal Linux processes. An example of this is given by the RT-Linux Sender application, where a user process defines a probe stream and a real-time task actually sends the packets on schedule, the two communicating via shared memory.

4. Active Probing Infrastructure

The basic components of an end-to-end active probing infrastructure are shown in figure 1. In each probing experiment, the Sender creates and transmits a probe stream which traverses some route in the network and terminates at the Receiver, a sink. Together with the probe sequence numbers available from the payloads, the packet arrival and departure time stamps define the raw outcome of the experiment. They are recorded by the Sender Monitor and Receiver Monitor respectively. Thus, although the task of the Sender is to send a probe stream with designed characteristics, it is the actual probe characteristics which constitute the experimental data and which are measured for use in analysis. Note that in the case of round-trip measurements the Sender and Receiver are two processes running on a single computer with a single Internet address.

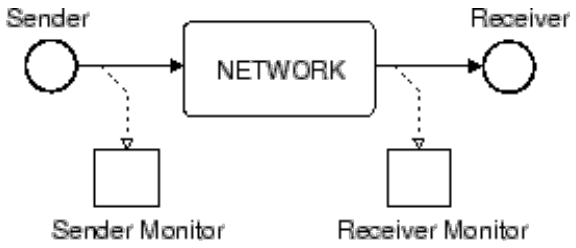

Figure 1: The basic components of an active probing infrastructure.

Each of the components above is a potential source of errors, which all involve timing in some form. These can be broken down further into errors due to reference timing sources (including synchronization issues), hardware issues such as interrupt cycles which drive process scheduling, and software issues including the performance of algorithms for software clock correction.

4.1 A High Precision Infrastructure

In existing systems timing problems of diverse origins, including inaccurate reference time sources, computer process scheduling, and lack of synchronization (in the case of one-way measurements), combine to produce time varying errors which can range from milliseconds to seconds. Each of these sources of error is quantified and addressed in the present system. The monitoring components at both sender and receiver are based on Global Positioning System (GPS) synchronized DAG3.2e [16] measurement cards with time stamping accurate to 100ns. The transmission component is based on a Real-Time Linux [17] Unix system transmitting over 100Mbps Ethernet, and is capable of delivering packets at an accuracy (using a 600MHz machine) limited by that of the interrupt latency in delivering packets to the card. For User Datagram Packet (UDP) probe packets varying from 30 to 1500 bytes this is uniformly below 10ms. The sender is capable of taking a specification of an arbitrary probe stream, given as a list of packet sizes and target inter-departure times, and of sending them consistently with this accuracy with almost no disruption due to kernel scheduling. In particular packets can be reliably sent back-to-back when required, an important dimension in probe stream design.

4.2 Comparison

Part of the attraction of active measurement has been its low cost, and ease of deployment, arising from the use of commodity equipment and software. Although we retain a commodity PC, Ethernet card, and freely available operating system software as the core of the system, clearly both of these advantages have been largely lost. However, the availability of a high accuracy reference system is an essential prerequisite to the evaluation of less complete solutions. A second goal of the project is to investigate the limitations of simpler systems. The variants considered are with or without GPS, using RT-Linux time stamping instead of the DAG, and using well designed sender programs in Linux and FreeBSD rather than RT-Linux. By bench marking against the full system, it is determined which combinations provide adequate performance for different tasks, allowing cheaper and therefore more numerous measurement nodes with limited but known precision.

To assess the accuracy of software based monitoring using the three operating systems we compared the time stamps generated by the receiver host against those from the DAG card, connected via passive Ethernet tap. (due to the high

performance of the DAG its current lack of filtering ability is not a limitation).

All the computers involved in the test were 500 MHz Pentium III based systems equipped with the same type of network interface card. The test hosts are connected in a star configuration to our switch using 100 Mbps Ethernet connections. Under FreeBSD and Linux the tcpdump utility was used to generate the time stamps, while under RT-Linux a real-time process performed this task.

The test packet stream consisted of 10000 packets each of 10 bytes with a lower end truncated exponentially distributed inter-departure time of mean 10 ms. This truncation is described further in the next section where similar test streams were used. For our purposes here the details of the stream are not important.

To better study timing errors arising in a single node we avoid metrics requiring end-to-end synchronization such as delay. We choose the inter-arrival time of the test packets as a basis for comparison. Over a 10 ms time interval, the mean inter-departure time of the test stream, the typical 50PPM clock skew would result in an average 0.5ms offset error, which is below the clock resolution of the compared host. Hence we do not use NTP but rely on the software clock, which is sufficiently accurate over these time scales.

There are two fundamentally different types of measurement errors. Those due to scheduling problems typically result in spikes in the order of ms. These appear when the operating systems performs a high priority task, which delays the processing of the arrived packet or the time stamp generation. The second group are due to the clock reading inaccuracies, which are typically in the range of ms.

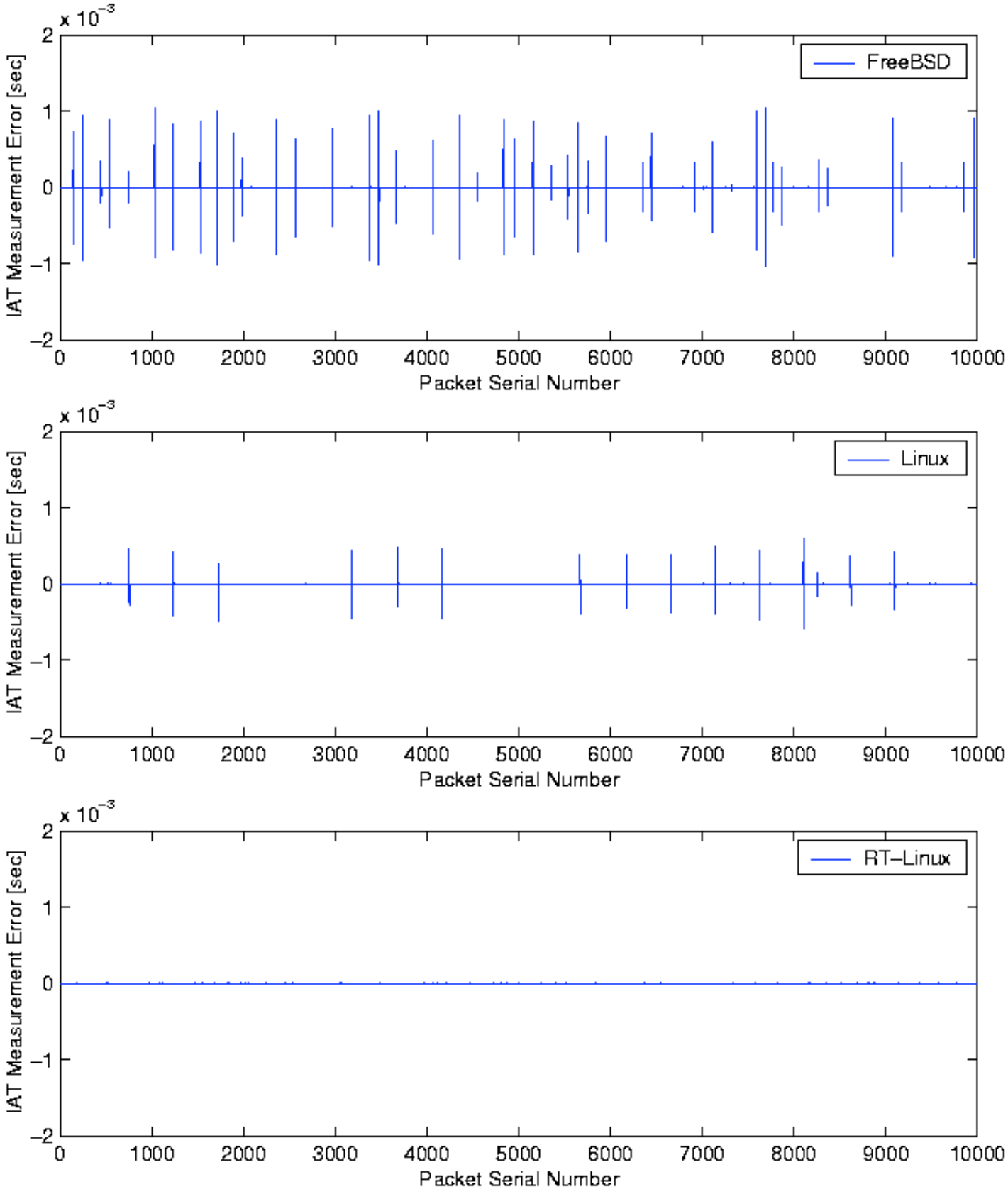

Figure 2. Monitoring comparison of BSD, Linux and RT-Linux comparing inter-arrival time errors

Comparing the results for RT-Linux to those for FreeBSD and Linux reveals the fundamental difference between multi-tasking and real-time operating systems. However, even the real-time system is limited by the constraints of the available hardware.

A number of experiments were performed to compare the suitability of the different operating systems for running an active probe sending application. All versions of the sender software use the so called *hybrid* timing [11]. In each case the

sending algorithm is implemented in two modules, the timing process running separately from the process generating the time series, the two communicating via shared memory. In the case of RT-Linux only the timing process is implemented as a real-time task.

As the measure of timing accuracy we use the difference of the targeted and measured packet inter-departure times. This choice is based on the same reasoning as in the monitor comparison. In each experiment 10000 packets were used of the same size.

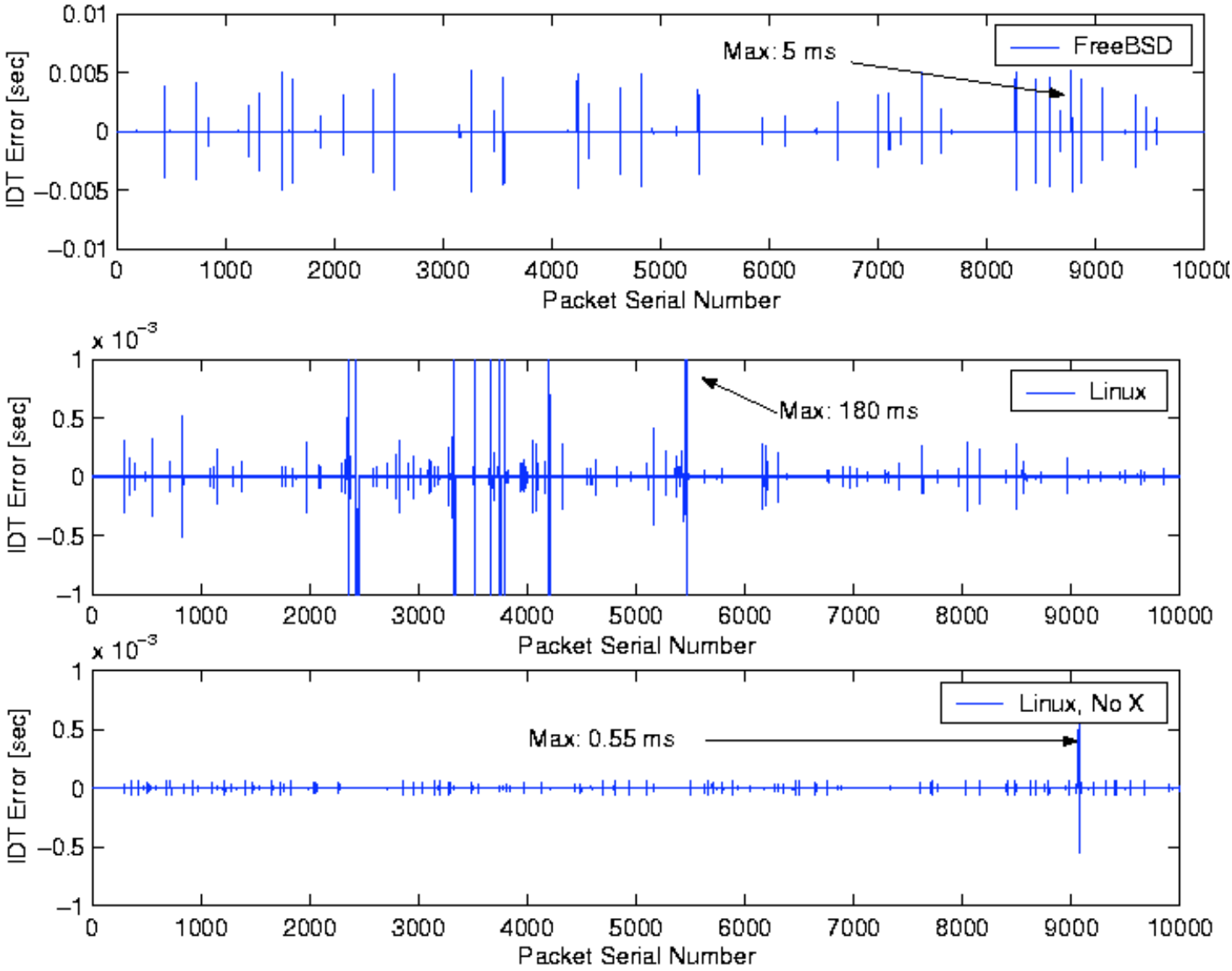

Figure 3. Sender comparison of BSD and Linux

The top graph of figure 3 shows the result of the experiment run with a sender on a FreeBSD host. This plot shows very similar characteristics to the results of the monitoring test (see figure 2). The delays due to scheduling problems appear as errors in the range of a couple of milliseconds. The schedulers of the Linux and FreeBSD kernels allow the root user to set the scheduling policy and priority of the executed processes using the sched_setscheduler system call. This helps to minimize the errors due to the execution of other processes. The results shown on the top graph of figure 3 are from a host, configured as an average client with a window manager and only a single user logged on, running the sending process as its only application. The same measurement was repeated using the sched_setscheduler system call to set maximum priority and to avoid pre-empting the execution of the sending process as much as possible. (The scheduling policy was set to SCHED_FIFO and its priority to sched_get_priority_max.) Running the sender with these changes on the FreeBSD host did not appreciably affect the measurement result compared to the application run on a minimally loaded system.

The experiment was repeated with a host running Linux, and the results shown in the bottom graphs of figure 3. The graph in the middle shows the results from a host running a window manager, serving a single user running a number of applications. The user changed the virtual screens displayed a couple of times during the measurement which resulted in spikes due to scheduling problems as high as 180 ms. The bottom graph shows the result of the experiment repeated on an idle host not running even a window manager, and running the sending process as the only active user application. The

effect of decreasing the system load is is dramatically demonstrated by the difference between these plots. The first Linux measurement was then repeated using the sched_setscheduler system call as in the case of FreeBSD. The results then became practically identical to those of the second experiment, decreasing the maximum error of 180 ms of the first experiment to below 1 ms.

The results of our tests demonstrate that the worst case scheduling accuracy in the case of the non real-time operating systems can exceed even 100's of milliseconds. However, with careful configuration and prioritization of the sender task the scheduling errors can be kept in the millisecond range. We have also observed that the frequency of these `ms' size errors, on our test machines running a very limited number of tasks, was in the range of tens per 10000 packets sent.

The RT-Linux sender essentially eliminates scheduling errors. The accuracy of the RT-Linux timing is limited by the properties of the hardware, determined mainly by the interrupt handling characteristics of the CPU, resulting in an upper bound on scheduling inaccuracies observed to be below 15ms on our 600 MHz Pentium III based test machines.

These tests provide information on the accuracy achievable on different platforms, however the final choice of Sender platform will depend on the requirements of the application and the nature of the metrics to be measured.

5. Applications

Active measurements are successfully used in a wide variaty of applications. The dominating application field of these measurement methods is the area of end-to-end performance measurements, benefiting from most of the advantages of the active probing. Besides the end-to-end performance metrics like latency and loss statistics, active measurements are used to measure availability and are also widely deployed in the area of topology discovery.

5.1 Example Application - Bandwidth Estimation

The benefits of an accurate infrastructure are apparent from figure 4, where many of the `visible queueing' lies in the millisecond range, which would be lost to noise in other systems. Estimates of metrics based on measuring these slopes would obviously suffer, and indeed many methods involved complex and heuristic aspects, involving much averaging, in an attempt to reduce the impact of `noise'. With the accurate infrastructure, rather than averaging, one can meaningfully examine high resolution histograms, which yields important new information.

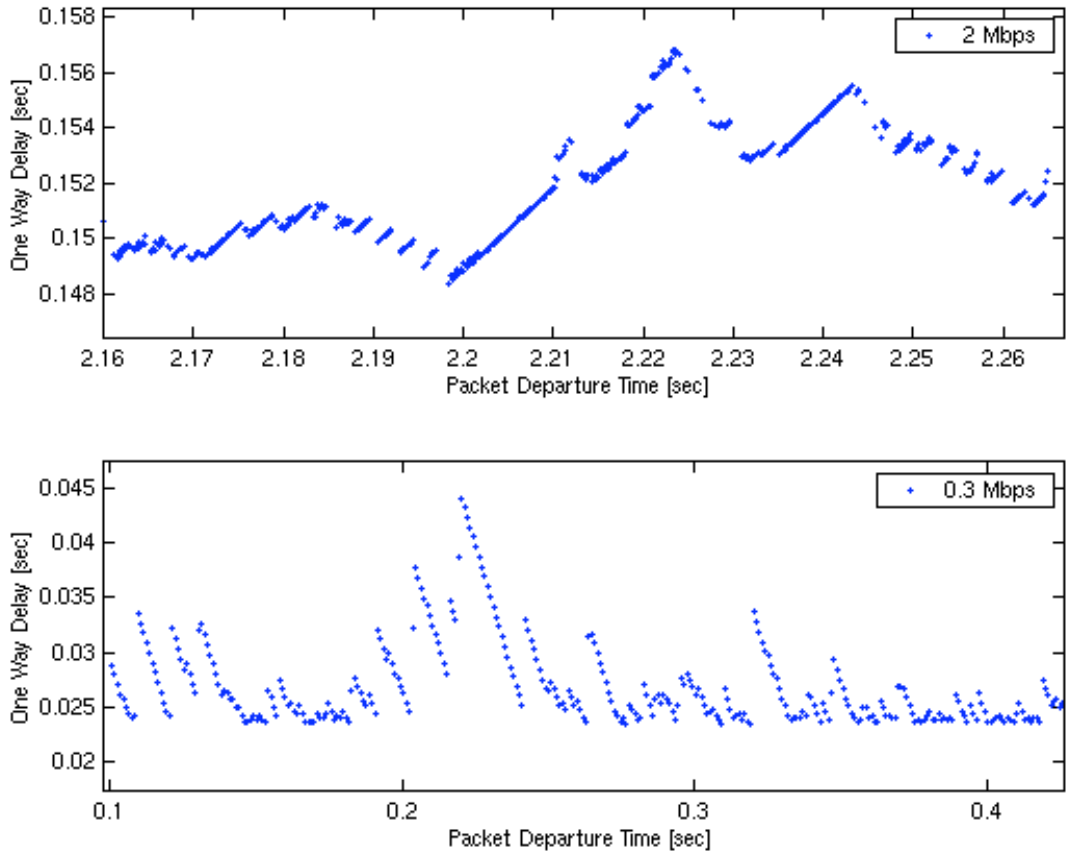

Figure 4. Two plots demonstrating the benefits of the high accuracy infrastructure - The delay plot of a probe stream with a transmission rate exceeding the bottleneck bandwidth (2 Mbps) compared to a stream remaining below it (0.3 Mbps)

The figures presented in this paper are the results of some initial one-way measurements over the Internet, between EMULab in Melbourne and the WAND group at the University of Waikato.

The measured route consisted of 13 hops, both the sender and the receiver residing on a 100Mbps LAN. It is also known that the bandwidth of the outgoing link from EMULab's gateway is 10Mbps and the bottleneck bandwidth of the route was somewhat below 2 Mbps. The results presented in this paper are based on a measurement setup using GPS synchronized DAG cards both as sender and receiver monitors and a RT-Linux based sender.

The plot of the lower rate stream on figure 4 clearly illustrates the effect of the background traffic on the one-way delay. On the arrival of background traffic bursts, or even just larger sized packets, the delay value jumps higher and then gradually falls back. This effect corresponds to a number of probe packets catching up to each other in a queue, then leaving the queue with a rate corresponding to the transmission rate of the given link. This effect is seen in on the lower plot of figure 4. As a result of this packet bunching, the slope of the delay decrease can be used to estimate the link bandwidth, as for example in [6]. Alternatively, similar information is carried in the inter-arrival time series. As is well known, the link rate determines the inter-packet times of the `spaced out' back-to-back packets leaving it, which when appearing as inter-arrival times at the receiver can be used to estimate the link bandwidth. Methods described in [6,26,27,28] are all based on this fundamental property.

The higher rate streams saturate the route and experiences loss. The upper plot of figure 4 shows a close up of the 2Mbps stream. Packet losses lead to decreased delay for packets immediately following the lost one (as the queue size at the router with loss is smaller), while consecutive arrivals with a steady rate exceeding the bottleneck rate leads to linear increase in the delay. The slope of the increase is again determined by the link bandwidth in a simple way.

Figures 5 and 6 demonstrate the importance and benefits of probe stream design and of high accuracy, high resolution active probing.

Two very different streams are used in this experiment. The 300kbit rate stream is a simple periodic one of 1500Byte packets with constant 40ms inter-departure time. Figure 5 shows the phase plot and the histogram of the one-way delay and it's differenced series, the delay variation experienced by this stream.

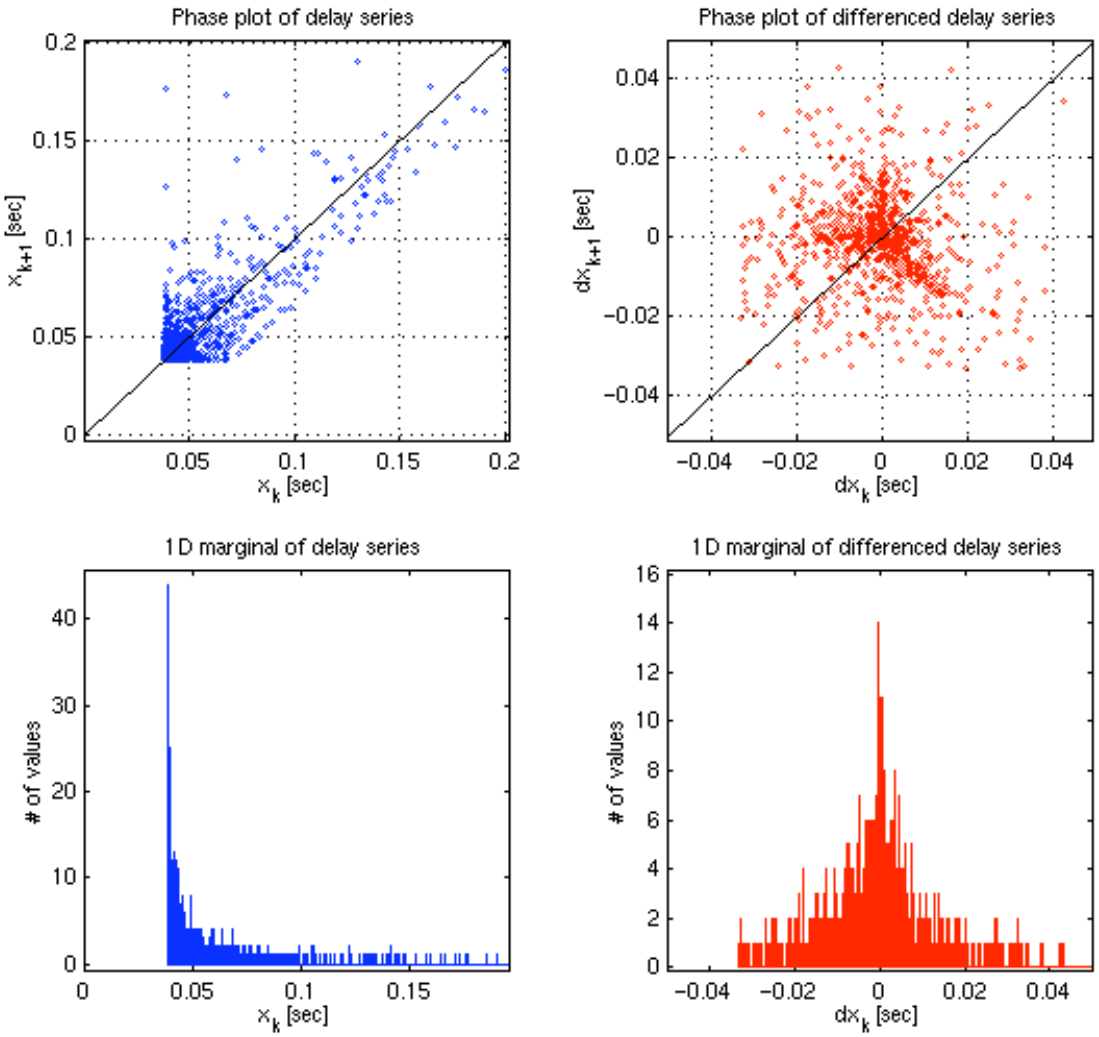

Figure 5. Delay and delay variation phase plot and histogram of the periodic stream

The second stream is a simple `designer' stream constructed from packet pairs where the inter-departure time within the pair is a constant 130ms, corresponding to back-to-back packets on the outgoing 100Mbps link, and pairs are independently separated according to an exponential distribution with a mean of 2 seconds. As the packet size used in the experiment is 1500 bytes the average rate of this stream is only 18kbps. Sending the packet pairs ensures that many packets will queue behind each other in queues along the route, whereas the periodic stream has to rely on background traffic to create this effect. The difference is shown very clearly in the delay variation histograms on figure 5 and 6. The 18kbps stream's histogram shows clearly a peak corresponding to the of packets spaced out by the bottleneck link while the same information is much less apparent from the histogram of the periodic stream. The phase plots of figure 6 clearly demonstrate the deterministic effect of the bottleneck on the delay characteristics while the plots of figure 5 are dominated by the random effects of the background traffic.

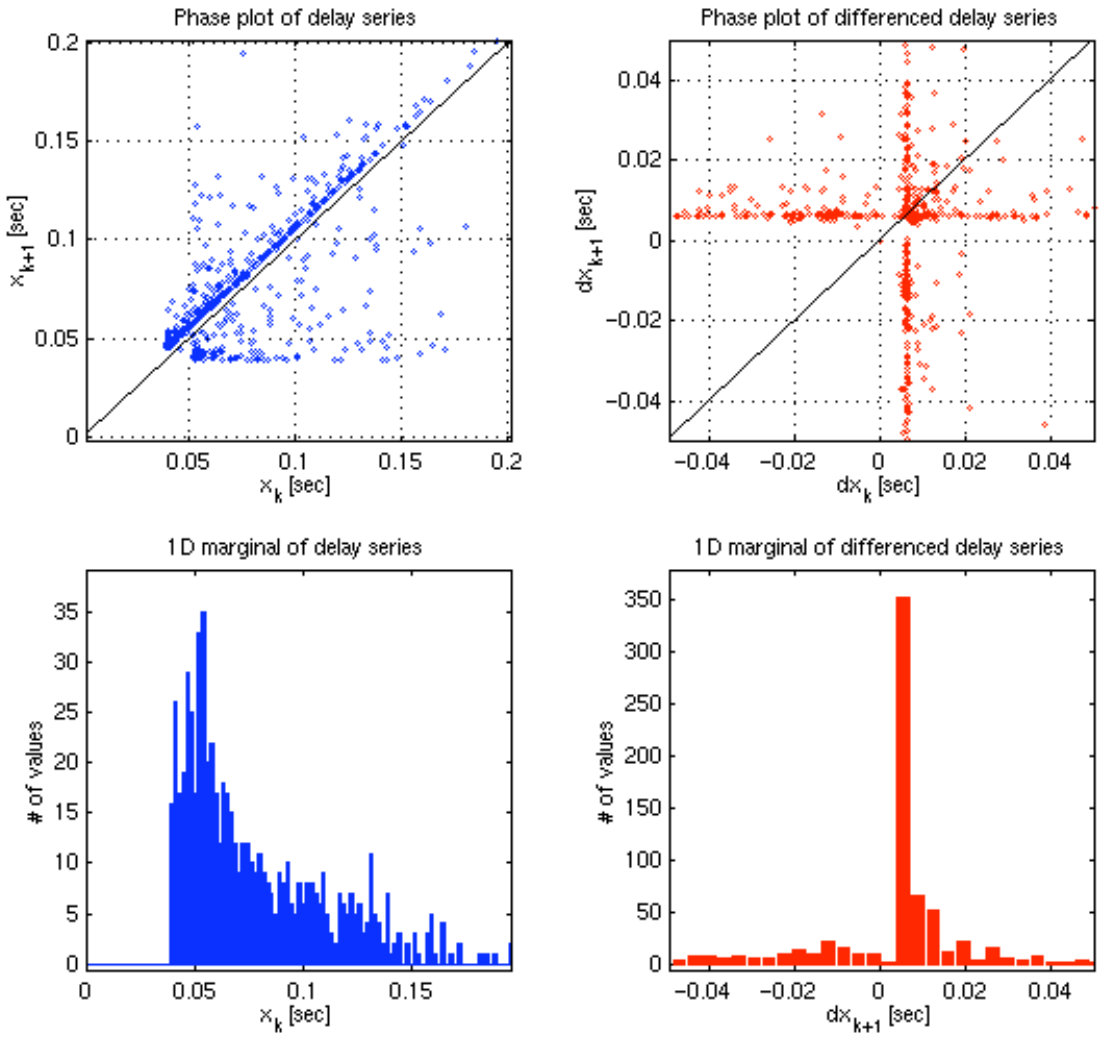

Figure 6. Delay and delay variation phase plot and histogram of the 'designer' probe stream

While the results shown on these figures are based on traces from the highly accurate and GPS synchronized DAG card monitors and an RT-Linux based Sender, the results of the same measurement repeated using only Linux applications for each task showed no qualitative differences compared to those shown above. This is the result of using inter-arrival time as a metric, and thereby not requiring the synchronization of the measurement hosts, and controlling the measurement errors via careful design of the infrastructure implementation.

This simple example demonstrates the benefits of using specially designed probe streams for active measurements. Bandwidth measurements proposed by [27,26] rely on sending probes back-to-back if possible while the methods proposed in [10,18] require precise control of the inter-departure times between the probe packets. These developments in the active probing methodology require more and more control over measurement errors and emphasize the importance of a high precision, reliable and simple measurement infrastructures. Our future work is focusing on the development and analysis of new measurement methods based on high resolution probing and benefiting from the accuracy of the new measurement infrastructure.

6. Conclusion

An overview of Internet measurement was presented, contrasting traditional passive and network core measurement techniques from the more end-to-end oriented *network edge* measurement, which is generally performed actively. The advantages of the active network-edge combination were given, motivating a detailed discussion of active measurement.

The difficulties inherent in active measurement on inexpensive PC based systems were discussed, in particular timing difficulties. To evaluate the performance of various alternatives, a highly accurate reference active probing infrastructure was developed, allowing the reliable measurement of network wide delays, and the reliable transmission of high resolution active probe streams, with well under millisecond accuracy. The system employs DAG card based monitors and a flexible RT-Linux based probe sender application.

After comparing different Unix on PC based solutions for the sender component of the infrastructures, RT-Linux was found

to be clearly superior, controlling scheduling errors down to 10 microseconds, whereas Linux and FreeBSD systems have errors in the millisecond range. However, carefully designed sender software and procedures can reduce these errors significantly, to the point where they are viable for measurements of some metrics which do not require end-to-end synchronization. Network based synchronization using NTP was examined in detail and was found to produce highly undesirable effects in the sub-millisecond range, and is not recommended.

Finally the reference infrastructure was used to perform preliminary experiments over an Internet route, illustrating the clarity with which queueing can be visualized with the reference system. Finally the benefits of using low rate `designer probe' streams as an efficient means of measuring link rates was discussed and illustrated.

Acknowledgments

The authors wish to thank Andrew Anderson and Michael Dwyer at EMULab for their invaluable expertise in Unix systems, and to David Hornsby for his assistance in accessing GPS. The generous support of the WAND group at the University of Waikato, particularly that of Jörg Micheel and including the loan of a DAG3.2e, is warmly acknowledged. This work was supported by Ericsson.

References

[1] L. Cottrell and W. Matthews, ``*Comparison of surveyor and ripe*,'' SLAC -http://www.slac.stanford.edu/comp/net/wan-mon/surveyor-vs-ripe.html, March 2000.

[2] S. Kalidindi and M. J. Zekauskas, ``*Surveyor: An infrastructure for internet performance measurements*,'' INET'99 -http://telesto.advanced.org/ kalidindi/papers/INET/inet99.html, June 1999.

[3] H. Uijterwaal and O. Kolkman, ``*Internet delay measurements using test traffic: Design note,*'' Tech. Rep. RIPE-158, RIPE NCC -http://www.ripe.net/ripencc/mem-services/ttm/Notes/RIPE_158.ps.gz, June 1997.

[4] T. McGregor, H.-W. Braun, and J. Brown, ``*The nlanr network analysis infrastructure,*'' IEEE Communications Magazine special issue on "Network Traffic Measurements and Experiments", http://wand.cs.waikato.ac.nz/wand/publications/, May 2000.

[5] W. Matthews and L. Cottrell, ``*The pinger project: Active internet performance monitoring for the henp community,*'' IEEE Communications Magazine special issue on "Network Traffic Measurements and Experiments", http://www-iepm.slac.stanford.edu/paperwork/ieee/ieee.ps.gz, May 2000.

[6] J. C. Bolot, ``*Characterizing end-to-end packet delay and loss in the internet,*'' Journal of High-Speed Networks, vol. 2, pp. 305-323, September 1993.

[7] V. Paxson, ``*End-to-end internet packet dynamics,*'' in Proceedings of ACM Sigcomm'97, (Cannes, France), pp. 139-152, September 14-18 1997.

[8] V. Jacobson, *Pathchar - a tool to infer characteristics of Internet paths*, available at: http://www.employees.org/bmah/software/pchar/ ed., 1997.

[9] K. Lai and M. Baker, ``*Measuring bandwidth,*'' in Proceedings of IEEE Infocom'99, (NY, NY), March 21-25 1999.

[10] B. Melander, M. Björkman, and P. Gunningberg, ``*A new end-to-end probing and analysis method for estimating bandwidth bottlenecks,*'' in Proceedings of Globecom'00, (San Francisco), November 2000.

[11] J. Andren, M. Hilding, and D. Veitch, ``*Understanding end-to-end internet traffic dynamics,*'' in IEEE GLOBECOM'98, vol. 2, (Sydney, Australia), pp. 1118-1122, November 1998.

[12] J.-C. Bolot and A. Vega-Garcia, ``*The case for FEC-based error control for packet audio in the internet,*'' ACM/Springer Multimedia Systems, 1999.

[13] S. Moon, J. Kurose, P. Skelly, and D. Towsley, ``*Correlation of packet delay and loss in the internet,*'' Tech. Rep. TR 98-11, Department of Computer Science, University of Massachusetts, January 1998.

[14] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, ``*Measuring and modeling of the temporal dependence in packet loss,*'' in Proceedings of IEEE Infocom'99, (NY, NY), March 21-25 1999.

[15] S. B. Moon, P. Skelly, and D. Towsley, ``*Estimation and removal of clock skew from network delay measurements,*'' Tech. Rep. 98-43, Department of Computer Science, University of Massachusetts at Amherst, 1998.

[16] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson, ``*Design principles for accurate passive measurement,*'' in PAM 2000, The First Passive and Active Measurement

Workshop,http://pam2000.cs.waikato.ac.nz/final_program.htm, (Hamilton, New Zealand), April 2000.

[17] V. Yodaiken, ``*The rtlinux manifesto,*'' tech. rep., Department of Computer Science, New Mexico Institute of Technology, 1999.
available at http://www.rtlinux.org.

[18] V. Ribeiro, M. Coates, R. Riedi, and S. Sarvotham, ``*Multifractal cross-traffic estimation,*'' in Proceedings of the ITC Specialist Seminar on IP Traffic Measurement, Modelling and Management 2000, (Monterey, CA), September 18-20 2000.

[19] L. Cottrell, ``*Comparison of some internet active end-to-end performance measurement projects,*'' SLAC -http://www.slac.stanford.edu/comp/net/wan-mon/iepm-cf.html, July 1999.

[20] D. Mills, ``*The network computer as precision timekeeper,*'' in Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting, (Reston VA), December 1996.
pages 96-108.

[21] P. Abry, D. Veitch, and P. Flandrin, ``*Long-range dependence: revisiting aggregation with wavelets*,'' Journal of Time Series Analysis, vol. 19, pp. 253-266, May 1998.

[22] D. Mills, ``*Precision synchronization of computer network clocks,*'' Tech. Rep. Report 93-11-1, Electrical Engineering Department, University of Delaware, November 1993.
66 pages.

[23] D. Mills, ``*Internet time synchronization: the network time protocol*,'' IEEE Trans. Communications COM, vol. 39, pp. 1482-1493, October 1991.
Condensed from RFC-1129.

[24] A. McGregor and H.-W. Braun, ``*Balancing cost and utility in active monitoring: The amp example,*'' NLANR -http://wand.cs.waikato.ac.nz/wand/publications/inet2000-amp-html/paper.html , February 2000.

[25] M. Aron and P. Druschel, ``*Soft timers: efficient microsecond software timer support for network processing,*'' Operating Systems Review, vol. 34, pp. 232-246, December 1999.
Also in 17th ACM Symposium on Operating System Principles (SOSP'99).

[26] K. Lai and M. Baker, ``*Measuring link bandwidths using a deterministic model of packet delay*,'' in Proceedings of IEEE Infocom'00, (Tel Aviv), March 2000.

[27] R. Carter and M. Crovella, ``*Measuring bottleneck link speed in packet-switched networks,*'' Tech. Rep. 1996-006, Department of Computer Science, Boston University, 15, 1996.

[28] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*.
PhD thesis, U.C. Berkeley, 1997.
ftp://ftp.ee.lbl.gov/papers/vp-thesis/dis.ps.gz.

[29] K. Claffy., *Internet measurement and data analysis: topology, workload, performance and routing statistics*.
NAE '99 workshop, CAIDA -http://www.caida.org/outreach/papers/Nae/, 1999.