Survey on the End-to-End Internet Delay Measurements

Junfeng Wang[3,1], Mingtian Zhou[1], and Yuxia Li[2]

[1] College of Computer Science & Engineering
University of Electronic Science & Technology of China
mailwangjf@yahoo.com.cn, mtzhou@uestc.edu.cn
http://www.ccse.uestc.edu.cn

[2] Graduate School, Chongqing Communication Institute
liyuxia20022003@yahoo.com.cn

[3] Institute of Software, Chinese Academy of Sciences
http://www.ios.ac.cn

Abstract. The end-to-end delay of Internet is a fundamental indicator for network performance evaluation and has been becoming a hot issue in network measurements in recent years. There are two kinds of metrics related to the end-to-end latency, i.e., Round Trip Delay and One-way Delay metric. In the paper, we survey the recent progresses on these two types of metrics measurement. Special concerns are on the clock synchronization issue in one-way delay measurement. The problems in deploying end-to-end delay measurement and the timestamping issue are also summarized. The potential new research directions in delay measurement and analysis are discussed.

1 Motivations

The unexpected explosion of the Internet, its usage to deliver increasingly important and various services make network monitoring and performance measurements essential for effective network management. Many applications may benefit from the knowledge of the end-to-end delay metrics. This section describes the motivations for measuring the end-to-end latency of the Internet.

Network latency is an important indicator of the operating network status, which changes with the variations of the network traffic patterns and congestions. Many QoS sensitive applications (VoIP, Stream Media for instance) require the delay constraints to be met. Therefore, the knowledge of the end-to-end delay can be used for Service Level Agreement (SLA) validation between network service providers and customers or between neighboring network service providers [1, 2]. Network operators should take corresponding actions to guarantee the QoS parameters for services if delay requirement fails[3, 4].

Delay metrics are also the foundations for many other metrics measuring, such as bandwidth, jitter, and packet loss measurements. *pathload* employs one-way delay measurement to estimate the end-to-end available bandwidth[5]. *pathchar* uses end-to-end round-trip delay to estimate the per-hop capacity

Z. Mammeri and P. Lorenz (Eds.): HSNMC 2004, LNCS 3079, pp. 155–166, 2004.

and link latencies[6]. Carter introduced the packet pair technique and developed *bprobe* to measure the end-to-end capacity from the difference of end-to-end round-trip delays(dispersion) experienced by consecutive back-to-back probe packets[7]. Siegell employed one-way delay to infer the network topology in[8]. IETF IP Performance Metrics Working Group (IPPM-WG) defined two basic delay metrics, the Round-trip Delay metric[9] and the One-way Delay Metric[10]. The derived metrics such as IPDV[11] and packet loss patterns[12] are also based on above one-way or round-trip delay measurement.

In addition, through the end-to-end delay measurements, researches can learn more about the underlying properties or characteristics of the current networks, for example, network topology, traffic patterns, protocol distributions, etc. The knowledge in turn can be feeded into measurement-based simulation systems to guide the capacity planning, application tuning and performance lifting.

1.1 Round-Trip Delay and One-Way Delay

The end-to-end delay measurements include Round-trip delay and One-way Delay measurements respectively. Round-trip delay measurement of `TCP/IP` networks may be dated back to 1971 on the ARPANET[13], while in December 1983, Mike Muuss developed the famous utility *ping* to diagnose the network connectivity from the end-to-end round trip delay[14]. *ping* utilizes the Echo `request/reply` mechanism defined in the Internet Control Message Protocol (ICMP) to accomplish the end-to-end connectivity measurement[15]. It has been an indispensable tool in network environments. Its IPv6 implementation, which commonly referred to as *ping6* has also been developed in many systems.

The principle of *ping* is simple and straightforward, the drawback of which is that most routers or gateways may disable the ICMP `request/reply` function for performance or security reasons. Hence, other *ping-like* utilities emerge, such as the *TCP ping* which employs the `SYN/ACK` mechanism in handshaking procedure of Transmission Control Protocol (TCP)[16, 17].

End-to-end one-way delay measurement also uses nearly the same working mechanism as that of round-trip delay. Measuring the one-way delay instead of the round-trip delay is motivated by following reasons[10]:

- In current Internet, there may exist asymmetric paths, the performance of forward and reverse path between peer systems may be different.
- Even when the two paths are symmetric, the different and asymmetric queuing policies in routers may result in different performance characteristics.
- At the same time, some other applications, FTP or Video on Demand (VoD) for example, are more concerned on the unidirectional performance.

Above one-way and round-trip delay measurements are also fell into the active measurement. They inject extra traffic (probe packets) into measured networks, which will disturb the properties of the carried traffic on the network in some measure. Measuring one-way delay passively does not generate additional traffic into underlying network, but need to keep track the departure and arrival times for all probe packets at measurement points and should communicate with each other to correlate the timestamps for delay calculation. Passive measurement model will generate significant amount of traffic for transmission measurement data and thus rarely deployed[18].

In this paper, we survey the state of the art of end-to-end delay measurements in recent years. We discuss the key issues both in the context of active and passive measurements. For simplicity, we denote end-to-end round-trip delay and end-to-end one-way delay by RTD and OWD respectively. The rest of paper is organized as follows. In section 2, we reviews those important projects that perform RTD and OWD measurements. The international organizations, which involved in RTD and OWD standardization are also summarized. Section 3 presents RTD measurement. Section 4 concerns on OWD measurement and discusses the end system clocks synchronization algorithms. In section 5, we focus on two key issues in delay measurements, i.e, timestamping and the probing time interval selection both in RTD and OWD measurements. Lastly, section 6 discusses the trend in delay measurements and concludes the paper.

2 Important Projects and Organizations

Many projects have involved in delay measurements. Based on Paxson's Network Probe Daemon system[19], the National Internet Measurement Infrastructure (NIMI) project is designed to construct a worldwide, distributed, and scalable measurement infrastructure. *zing* utility is integrated to perform active delay measurement[20]. Surveyor is an active network measurement infrastructure that is being currently deployed at participating sites around the world. It implements the measurements proposed by IETF IPPM Working Group. Surveyor achieves OWD measurement by sending 40 bytes UDP packets scheduled according to a Poisson Process with average rate 2 packets per second[21]. The Internet End-to-end Performance Measurement (IEPM) project developed PingER to measure the end-to-end network performance including RTD. It simply uses *ping* to carry out the long-term RTD measurement and has been deployed in 79 countries[22, 23]. RIPE (Reseaux IP Europeens) Test Traffic measurement project is to ensure the administrative and technical coordination necessary to enable a pan-European IP network OWD and other metrics measurements[24]. Other famous performance measurement projects include the NALAR Active Measurement Program (AMP)[25] and the Sprint IP Monitoring (IPMON) infrastructure[26]. The former provides active RTD measurement across America, while the latter collects the GPS synchronized packet level traces from the Sprint Internet backbones for network delay performance evaluation[27].

Efforts toward the standardization of RTD and OWD metrics measurements and evaluation are mainly carried out within the framework of the IP Performance Metrics Working Group (IPPM-WG) of the IETF. Proposed standards include one-way delay[10], round-trip delay[9], and other related metrics, e.g., connectivity[28], one-way packet loss[12] and IPDV[11].

3 Round-Trip Delay Measurement

The RTD measurement often takes the form of active measurement. It injects specified length of probe packets into network and receives corresponding response from the intended destination. The differences between a probe's arrival time and its departure time at sender is reported as the round trip delay. Though RTD can only provide a coarse illustration of network performance compared with the one-way delay measuring, its ease of deployment and a variety of well-known approaches make RTD measurement still attractive in many circumstances.

Definition 1. *An active RTD measurement is a 8-tuple* $RTD = \{P, A, D, L, M, S_T, R_T, t_0\}$*, where:*

(1) P is a set with finite sequence of packets, and let $p(i)$ denote the *i-th* packet in P.
(2) A is a set with non-empty end system addresses.
(3) D is a address function, which is defined as: $D : P \to A \times A$. Function D determines the two end-to-end systems on which to perform the delay measurement.
(4) L is a length function that is defined as: $L : P \to N$. This function specifies the packet length for each probe packet in active measurement. Commonly, in a measurement, the length of all probe packets is the same.
(5) M is a finite set with non-empty measurement methodologies. Now typical methods include ICMP `request/reply` method, TCP-based `SYN/ACK` method and UDP-based method, etc. As the packet length, only one method is bound to all packets in a measurement case.
(6) S_T is a monotonic increasing function, which is defined as: $S_T : P \to R$. The S_T determines when a packet is to be sent to its potential receiver or responder. The widely used methods are to send probes periodically as *ping* or according to Poisson Process as *zing*[29].
(7) R_T is a timestamping function for all response packets: $R_T : P \to R$. R_T reflects the network performance or the traffic pattern when the probe packet is traversing the end-to-end path. It determines the receive time of probe responses at the sender side.
(8) $t_0, t_0 > 0$ is a time out parameter for packet loss decision. If the calculated RTD of packet $p, R_T(p) - S_T(p)$ is greater than t_0, then the packet p is deemed as lost during measurement.

Above definition is similar to the "Type-P" description in [9]. but with little variation, it can be adopted to characterize the OWD measurement. The characteristic of self-synchronization makes the RTD calculation much easer.

$$RTT(i) = \min\{R_T(p(i)) - S_T(p(i)),\ t_0\} \tag{1}$$

Equation (1) illustrates the RTD of the *i-th* probe packet. The RTT with value t_0 indicates that the packet loss occurs. Definition 1 makes no assumptions on the stability of routes between end systems or on the `symmetry/asymmetry` of paths. Therefore, it is unreasonable to deduce the OWD from RTD measurement. Rigorous assumptions on the above two aspects could make OWD estimation easer based on RTD.

4 One-Way Delay Measurement

One-way Delay (OWD) measurement can be achieved by Non-intrusive (passive) and intrusive (active) measurements. Passive OWD measurement requires generating a unique packet identity and timestamp for each captured packet at end systems (i.e., observation points), and then transferring the packet identities and timestamps to calculate the OWD. Zseby shows a preliminary model for passive OWD measurement as shown in Fig.1[1].

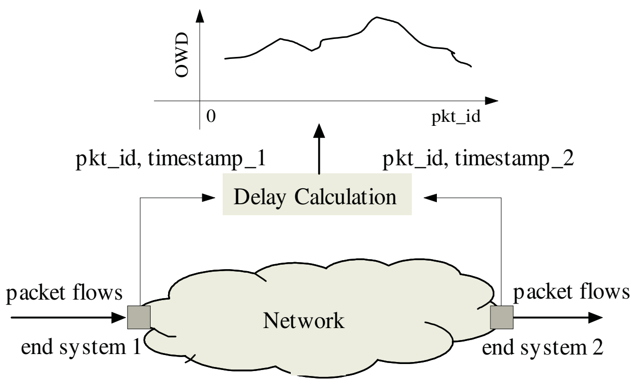

Fig. 1. The passive OWD measurement model

It is clear that to measure the OWD passively, the clocks of end systems must keep synchronized and how to sample the packets from network traffic should be considered too. The two issues also exist in active measurement. We address the synchronization problem in this section and deal with the sampling issue in next section.

4.1 Measuring OWD from Two-Way Measurement

An alternative way to measure OWD without clocks synchronization is to utilize the RTD measurement with the symmetry assumption that the OWD of the forward path (from the source to the destination of the probe packets) and the reverse path (from probe's destination to source) are the same. By halving the RTD, we could obtain the estimation of OWD.

Ciuffoletti adopted this assumption to measure clock offset and jitter asymmetry in [30] using UDP probing datagram, while Paxson employed TCP probing packets to access relative clock offset between two end systems under the same assumption[31].

4.2 Measuring OWD from One-Way Measurement

In Definition 1, if R_T is changed to denote the timestamping function for probe packets at their destination, then the raw OWD (ROWD) can be calculated as Equation (2):

$$ROWD(i) = \min\{R_T(p(i)) - S_T(p(i)),\ t_0\} \tag{2}$$

By subtracting the clock offset $\delta(i)$ from ROWD, the OWD is obtained:

$$OWD(i) = ROWD(i) - \delta(i) \tag{3}$$

If end systems are synchronized, the clock offset between clocks equals to zero, thus OWD can be deduced from Equation (2) directly.

The Global Positioning System (GPS) and Network Time Protocol (NTP)[32] are widely used in network time synchronization. In the context of delay measurements, their drawbacks also exist.

The GPS solution requires GPS receivers to be installed in end systems. It can provide reliable clock synchronization with high accuracy in the order of tens to hundreds of nanoseconds. As GPS receivers receive satellite signals, which require an unobstructed view of the sky, thus their deployment is limited because of the environment factors. Cost is another limitation for GPS applications in large-scale network performance measurements.

The approach used by NTP to achieve reliable time synchronization is to synchronize the clock to the time of the hierarchically-structured NTP servers. The NTP servers in lower levels obtain reference time from their predecessors. The root server gets time from external time sources, such as the Universal Coordinated Time (UTC) though GPS or other means. Because the NTP packets are distributed through in-band traffic, the clock synchronization errors are in the same order as the network latency. Nowadays, the most popular external clock synchronization solutions are still based on GPS.

However, Paxson argues that even endowed with GPS receiver, the ultimate clock readings derived from it remains prudent[31]. Therefore, many researches are concentrated on how to remove the clock offset from the raw OWD according to Equation (3) in the condition of non-external clock references.

Definition 2. *A clock C is a piecewise function that is differentiable except on a finite set of points. Let C_s and C_r denote the clocks of probes' sender and receiver. The difference between the times reported by a clock C and the* TRUE *clock is defined as* clock offset. *The offset of C_s relative to C_r at time t, $t \geq 0$ is $C_s(t) - C_r(t)$.*

Definition 3. *The* clock skew *is defined as the frequency difference of a clock C and the* TRUE *clock at time t. The skew of C_s relative to C_r is $\frac{dC_s(t)}{dC_r(t)}$, $t \geq 0$.*

Based on above definitions, there are two kinds of models in the effort to remove the clock offset, i.e, the mono-segment model and multi-segments model.

4.2.1 Mono-segment Model

The mono-segment model removes the *clock offset* δ based on the assumptions that there are no abrupt clock adjustments occurred and the *clock skew* remains constant in the whole measurement duration. Therefore, the raw OWD presents a steady increase or decrease trend if plotted in plan as shown in Fig.**??**. Currently proposed algorithms are focus on estimating the slope α of this steady trend. Moon developed a more robust algorithm called Linear Programming Algorithm (LPA) from one-way probing[33]. The scheme is depicted as:

$$(\hat{\alpha}, \hat{\beta}) = \arg_{\alpha,\beta} \min\{\sum_{i=1}^{N}[\Delta t_{rs}(i) - \alpha\tilde{t}_r(i) - \beta]\} \tag{4}$$

$$\text{subject to} \quad \Delta t_{rs}(i) - \alpha\tilde{t}_r(i) - \beta \geq 0$$

where $\tilde{t}_r(i) = R_T(p(i)) - R_T(p(1)),$ and $\Delta t_{rs}(i) = \tilde{t}_r(i) - \{S_T(p(i)) - S_T(p(1))\}.$ α and β are the slope and initial *offset* respectively as shown in Equation (4) (clock C_r is used for reference clock).

After α and β are estimated, we could synchronize C_s to C_r and calculate delay from the expression $\Delta t_{rs}(i) - \alpha\tilde{t}_r(i) - \beta$. Obviously, the values are not the true OWD, but rather the variable portion of OWD[33]. Paxson's skew estimation algorithm supposes that the forward and reverse one-way delays experienced by probe packets appear equivalent but opposite trends[19]. But this assumption does not hold as described previously. A comprehensive comparisons of the performance between LPA, Paxson's algorithm, linear regression algorithm, and piecewise minimum algorithm are also conducted in [33].

All the mono-segment algorithms are based on the assumption that the clock skew retains constant and with no clock adjustments in the course of measurement. In fact, computer clocks are sometimes subject to gradual or instantaneous adjustments. This fact results in multi-segments model in clock skew removal.

4.2.2 Multi-segments Model

The clock dynamics can be divided into three types: clock adjustments, frequency adjustments and clock drift[32]. The notions under multi-segments model are that end systems clocks' time and clock frequency can undergo abrupt adjustments, but there are no clock drift in measurement as in mono-segment model. Fig.2 shows an observation with clock adjustments and skew variations in our one-way measurement.

Li Zhang proposed a convex hull based algorithm to estimate and remove the relative clock skew in the context of clock resets[34]. To detect clock resets and remove clock skew from one-way delay traces, [34] first utilizes a *divide-and-conquer* technology to identify the number of clock resets. Then an algorithm called *R-Resets* is used to find the best clock skew and the time at which clock resets occur. The overall complexity is $O(Nw) + O((N/R)^R NR)$, where w is the window size related to *divide-and-conquer* algorithm, R denotes the estimated number of clock resets.

The limitations of the convex hull based algorithm are summarized as follows:

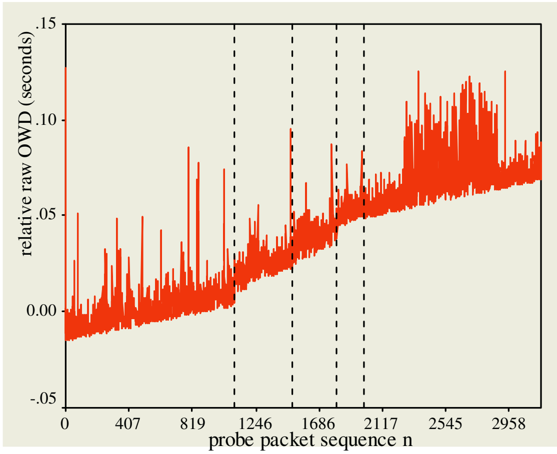

Fig. 2. An instance of raw one-way delay with multiple clock dynamics

- In identifying the number of clock resets, the *divide-and-conquer* algorithm assumes that clock resets do not happen very often and the minimal distance between consecutive clock resets is prior known. A tolerance level for comparing the slopes of two skew lines must be defined, which will significantly affect the ultimate number of clock resets.
- The *divide-and-conquer* algorithm also supposes that three consecutive intervals divided from delay trace contain at most one occurrence of clock reset. How to select a sound window size w is difficult in practice.
- After the number of clock resets is determined, *R-Resets* is used to estimate the best clock skew and the time when a clock reset occurs. The assumption behind this procedure is that the skew line slopes of clock stable periods are the same, e.g. clock skew does not change before and after a clock reset.
- An enhancement of the algorithm to detect the clock skew changes might not generate an optimal solution in some situations, which is also pointed out in that literature.

To detect the clock dynamics whether clock speed changes or clock resets, and address above limitations, a more general clock analysis model is developed by Junfeng Wang in [35] based on unidirectional delay time series segmentation technology (a special instance of clustering analysis). It segments the time series into multiple segments with each segment representing a stable period of the defined relative clock model. For a time series segment, clock skew can be estimated and removed by the Linear Programming Algorithm or convex hull algorithm to achieve relative clock synchronization between end-to-end systems. Therefore, it is also a two-stage algorithm. The first stage decides the potential times of clock activity changes, in which new cost and objective functions are introduced to partition and identify the optimal internally homogeneous segments in the time series. The new cost function $COST(TS)$ for time series TS is defined as Equation (5):

$$COST(TS) = \min\{\sum_{i=1}^{k}\sum_{j=a_i}^{b_i}[\Delta t_{rs}(j) - MSL(\tilde{t}_r(j) \mid i)]\} \quad (5)$$

where TS is the time series composed of P, k denotes the potential times of clock changes, and a_i, b_i are the candidate start and end points of the *i-th* segment. The *Maximal Skew Line (MSL)* is defined for each possible segment i as $MSL(i) = \sigma_i x + \mu_i$, while σ_i, μ_i subjected to Equation (6):

$$\begin{cases} \min\{\sum_{j=a_i}^{b_i}[\Delta t_{rs}(j) - (\sigma_i \cdot \tilde{t}_r(j) + \mu_i)]\} \\ \Delta t_{rs}(j) - (\sigma_i \cdot \tilde{t}_r(j) + \mu_i) \geq 0, \ a_i \leq j \leq b_i \end{cases} \quad (6)$$

Expression $MSL(\tilde{t}_r(j) \mid i)$ represents the expectation of $MSL(i)$ at $\tilde{t}_r(j)$.

The boundaries of each segment indicate an occurrence of clock dynamics. The computational complexity is of order $O(KN^2)$. Fig.2 also shows the clock dynamics positions with dotted vertical lines. A detailed investigation for the algorithm is also performed in [35].

Essentially, the time series segmentation algorithm is based on the LPA algorithm in [33]. As no rigorous assumptions on end system clock activities are introduced in the segmentation based algorithm, thus it could deal with more sophisticated clock dynamic situations than the LPA algorithm and the convex hull based algorithms. Furthermore, it is clear that the clock reset activity is only a special case of the general clock model[35]. The clock resets detection algorithm developed by Li Zhang in [34] can also be replace by the segmentation based algorithm in practice, but it is not true for the inverse cases, as the situation shown in Fig.2.

5 Key Issues in Measurements

5.1 Accurate Timestamping

Delay measurements are often carried out by the use of Internet hosts themselves to perform the measurements. The IP Performance Metrics Framework[36] proposes that the delay metrics should be deduced in terms of wire times instead of host times to avoid the errors introduced by hosts. Though the wire time is defined, it is not efficient to ensure that the wire times can be obtained in practice. In principle, timestamping is only possible when the wire time is associated to an event that is observable in host. But this statement never holds in practice[30].

Currently, most measurement software is developed with user-level C code written based on socket libraries and timestamps are marked in user space. Because of the process context swap, the end system load, `TCP/IP` protocol entities encapsulation/decapsulation, the timestamps suffer from uncertain deviation from the expected wire departure/arrival times. Kalidindi extensively discussed the time errors that may be yielded in each processing step from user space to wire or vice versa[37].

Timestamping in the operating system kernel provides high accuracy than in user level. Software-based solutions include libpcap library[38] and the Surveyor prototype implementation[37]. With the ever increasing link data rates, the demand for more precise and accuracy timestamping brings about the hardware-based methods. The DAG measurement card is an example of hardware-based timestamping with integrated GPS receiver for the clock synchronization[39].

5.2 Efficient Probing Schemes

Whether in active or passive delay measurements, the traces which contain the timestamping information are used to calculate delay metrics. The passive delay measurement must employ the sampling methodologies to capture packets and generate timestamps. In the context of active measurements, probe packets should be injected into network and the timestamps are recorded. In general, this is another form of sampling. How to select the time interval for the consecutive samples is crucial for reliable and effective measurements.

IPPM propose two sampling methods for trace collection, geometric sampling and Poisson sampling[37]. The common property of the two methods is of being unbiased and not predicable in advance. While in practice, the popular method is the periodic sampling because of simplicity such as those *ping-like* tools. These tools collect samples with predefined deterministic time intervals. [37] summarizes the drawbacks of periodic sampling to be partial observation and can perturb the traffic patterns.

6 Conclusions

In the paper, we survey recent efforts on end-to-end delay measurements. By formalizing the end-to-end delay measurements, we analyze those important factors and their progresses in a unified frame. In summary, the main concerns are focused on the clocks synchronization issue and proposed algorithms in the condition of without external clock synchronization mechanisms, and the accurate timestamping and efficient probing strategies.

As backbone bandwidth increases dramatically, measuring these core links directly will cause potential network performance degradation and safety threats, etc. More recently, much attention has been paying to the end-to-end delay measurements and then to infer the individual links performance through multicast-based or packet-pair technologies, such as Multicast-based Inference of Network internal Characteristics (MINC) project[40]. In parallel with this, end-to-end delay modelling is under its way. All these efforts will enhance the end-to-end delay measurements and network performance analysis, which will in turn benefit the design, development, control and management of the Internet and the Next Generation Internet(NGI).

References

1. Tanja Zseby. Deployment of Sampling Methods for SLA Validation with Non-Intrusive Measurements. *In Proceedings of the Passive and Active Measurement Workshop 2002 (PAM 2002)*, Fort Collins, Colorado, USA, March 2002

2. Zhiheng Wang, Amgad Zeitoun, Sugih Jamin. Challenges and Lessons Learned in Measuring Path RTT for Proximity-based Applications. *In Proceedings of the Passive and Acitve Measurement Workshop 2003 (PAM 2003)*, La Jolla, California, USA, April 2003
3. José Afonso, Vasco Freitas. Server Selection on the Internet Using Passive Probing. *In Proceedings of Multimedia Computing and Networking 1997 (MMCN 1997)*, San Jose, California, USA, February 1997
4. Ludmila Cherkasova, Yun Fu, Wenting Tang, *et al.* Measuring End-to-End Internet Service Performance: Response Time, Caching Efficiency and QoS. *HP Labs Technical Report*, HPL-2002-148, June 2002
5. Manish Jain, Constantinos Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *In Proceedings of the ACM SIGCOMM 2002*, Pittsburgh, PA, USA, August 2002
6. Allen Downey. Using Pathchar to Estiamte Internet Link Characteristics. *In Proceedings of the ACM SIGCOMM 1999*, Cambridge, MA, USA, September 1999
7. Robert Carter, Mark Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. *Performane Evaluation*, Vol. 27–28, Elsevier Science, pp.297–318, October 1996
8. Bruce Siegell, Joseph DesMarais, Mark Garrett, *et al.* Flix Project: Topology Discovery From One-way Delay Measurements. *In Proceedings of the Passive and Active Measurement Workshop 2000 (PAM 2000)*, Hamilton, New Zealand, April 2000
9. Guy Almes, Sunil Kalidindi, Matthew Zekauskas. A Round-trip Delay Metric for IPPM. IETF RFC 2681, September 1999
10. Guy Almes, Sunil Kalidindi, Matthew Zekauskas. A One-way Delay Metric for IPPM. IETF RFC 2679, September 1999
11. Carlo Demichelis, Philip Chimento. IP Packet Delay Variation Metric for IP Performance Metrics. IETF RFC 3393, September 1999
12. Rajeev Koodli, Rayadurgam Ravikanth. One-way Loss Pattern Sample Metrics IETF RFC 3357, August 2002
13. L.Kleinrock. Queueing Systems. Vol. 2: Computer Applications. Wiley-Interscience, New York, 1976
14. `http://www.sacpcug.org/eblue/0012/niobits.html`, March 2004
15. J. Postel. Internet Control Message Protocol. IETF RFC 792, September 1981
16. Ping vs. SYNACK. http://www-iepm.slac.stanford.edu/~cottrell/pinger/synack/ping-vs-synack.html, March 2004
17. J. Postel. Transmission Control Protocol. IETF RFC 793, September 1981
18. Tamás Varga, András Oláh. Quality of Service Monitoring in IP Networks by Using Two-way Active Measurements. *In Proceedings of the EUNICE 2000*, Enschede, the Netherlands, September 2000
19. Vern Paxson. Measurement and Analysis of End-to-End Internet Dynamics. Ph.D dissertation, University of California, Berkeley, 1997
20. Vern Paxson, Jamshid Mahdavi, Andrew Adams, *et al.* An architecture for large-scale Internet Measurement. *IEEE Communications*, Vol. 36, No. 8, pp.48–54, August 1998
21. Homepage of Surveyor, `http://www.advanced.org/surveyor`, March 2004
22. Les Cottrell, Connie Logg, I-Heng Mei. Experiences and Results from a New High Performance Network and Application Monitoring Tools. *In Proceedings of the Passive and Acitve Measurement Workshop 2003 (PAM 2003)*, La Jolla, California, USA, April 2003

23. Homepage of IEPM, `http://www-iepm.slac.standord.edu`, June 2003
24. Maximo Alves, Luigi Corsello, Daniel Karrenberg, *et al.* New Measurements with the RIPE NCC Test Traffic Measurements Setup. *In Proceedings of the Passive and Active Measurement Workshop 2002 (PAM 2002)*, Fort Collins, Colorado, USA, March 2002
25. Homepage of NLANR Measurement and Network Analysis Group, `http://moat.nlanr.net`, March 2004
26. Homepage of IPMON, `http://ipmon.sprintlabs.com/ipmon.php`, March 2004
27. Konstantina Papagiannaki. Provisioning IP Backbone Networks Based on Measurements. Ph.D dissertation, University of London, Feburary 2003
28. Jamshid Mahdavi, Vern Paxson. IPPM Metrics for Measuring Connectivity. IETF RFC 2678, September 1999
29. Yin Zhang, Vern Paxson, Scott Shenker. The Stationarity of Internet Path Properties: Routing, Loss, and Throughput. *ACIRI Technical Report*, May 2000
30. Augusto Ciuffoletti. Measuring One-way Metrics without a GPS. *In Proceedings of the Passive and Active Measurement Workshop 2002 (PAM 2002)*, Fort Collins, Colorado, USA, March 2002
31. Vern Paxson. On Calibrating Measurements of Packet Transit Times. *In Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, Madison, Wisconsin, USA, pp.11–21, 1998
32. David Mills. Network Time Protocol (version 3): Specification, implimentation and anlysis. IETF RFC 1305, March 1992
33. SUE B.Moon. Measurement and Analysis of End-to-End Delay and Loss in the Internet. Ph.D dissertation, University of Massachusetts Amherst, January 2000
34. Li Zhang, Zhen Liu, Cathy Honghui Xia. Clock Synchronization Algorithms for Network Measurements. *In Proceedings of IEEE INFOCOM 2002*, New York, USA, June 2002
35. Junfeng Wang, Mingtian Zhou, Hongxia Zhou. Clock Synchronization for Internet Measurements: A Clustering Algorithm. to appear in *Elsevier Science, Computer Networks*
36. Vern Paxson, Guy Almes, Jamshid Mahdavi, *et al.* Framework for IP Performance Metrics. IETF RFC 2330, May 1998
37. Sunil Kalidindi. Techniques to Scale the Surveyor Infrastructor. Surveyor Technical Report. STR-003, September 1998
38. Homepage of libpcap. `http://www.tcpdump.org`, March 2004
39. Jorg Micheel, Stephen Donnelly, Ian Graham. Precision Timestamping of Network Packets. *In Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement Workshop 2001*, San Francisco, California, USA, November 2001
40. Homepage of MINC. `http://www.research.att.com/projects/minc`, March 2004