Survey of network metrology platforms

Anne-Cécile Orgerie*†, Paulo Gonçalves*, Matthieu Imbert*, Julien Ridoux† and Darryl Veitch†
*ENS de Lyon - INRIA Rhône-Alpes - LIP (UMR CNRS, INRIA, ENS, UCBL)
46 allée d'Italie 69364 Lyon Cedex 07 - France
Email: annececile.orgerie@ens-lyon.fr, paulo.goncalves@inria.fr, matthieu.imbert@inria.fr
† Department of Electrical & Electronic Engineering - The University of Melbourne
Victoria 3010 - Australia
Email: jridoux@unimelb.edu.au, dveitch@unimelb.edu.au

***Abstract*—Internet services rely on communication networks to serve billions of end-users worldwide. Metrology platforms are used to assess the Quality of Service of such networks. This work reviews and classifies the existing metrology equipment and methods. In particular, we provide elements of comparison between different packet capture techniques and clock synchronisation methods, two essential building blocks of metrology platforms.**

***Keywords*- Clock synchronisation; packet timestamping; packet capture; network metrology platform**

I. Introduction

The ever-growing appetite of new applications for networking resources has not stopped since the Internet was built in the early-80s. In particular, emerging real-time applications such as voice over IP, online video games or financial market applications require high Quality of Service such as low latency and low jitter. That is why network service providers constantly measure the quality of their infrastructures by using dedicated metrology equipment.

Network measurements allow researchers and providers to obtain a comprehensive view on the usage and quality of service of networking infrastructures. Yet, the accuracy of these measurements greatly relies on network synchronisation performance. Indeed, the measured parameters are tightly linked to timestamping and timekeeping operations.

Accurate packet timestamping is a key challenge. First, the timestamping resolution should be fine enough to capture all the traffic. For instance, on a 10 Gbps link at full speed, if you are using timestamps with microsecond resolution, with 64-bytes packets (minimum packet size), you will have roughly 20 packets for the same timestamp value. In this case, the measurement equipment should have at least a 50 ns granularity in order to obtain different timestamps for every packet.

Secondly, networking devices are geographically distributed by nature and thus, synchronisation among them is hard to achieve over large distances. Tight synchronisation is however required to measure one-way latencies between two end-hosts for example. In order to obtain reliable metrology systems, it is essential to identify and consider all the sources that can influence the network measurement accuracy. Then, it is required to eliminate or precisely estimate (with models and experimental validations) these induced latencies to reach an acceptable level of synchronisation.

This paper aims at identifying the possible sources of inaccuracy in network monitoring tools, and to explore, classify and compare available network metrology platforms. Section II introduces the background. Section III reviews the available packet capture systems, while Section IV presents the clock synchronisation methods. Section V provides insightful comments to design an accurate and inexpensive metrology platform hence easy to deploy at a large scale. Section VI concludes this work and proposes future directions.

II. Background

Network measurement is a matter of concern since the emergence of network themselves. Commonly, three time-related parameters are measured to determine the QoS delivered by an infrastructure:

1) **throughput**: the average rate of successful data delivery over a given time slot, usually measured in bits per second;
2) **latency**: the time delay either one-way (from source to destination) or round-trip (from source to destination and back again);
3) **jitter** (or packet delay variation): the difference in end-to-end time delay between packets of a flow (any lost packets are ignored).

To compute (or more exactly measure) these parameters, people could put monitoring equipment on each network device (router or switch) of the path followed by packets to compute the time spent in each portion of the network. This solution would provide a precise view of the QoS of the whole network, but it is highly impractical, and prohibitive in terms of equipment time and required processing time for the obtained data. Commonly, the monitoring equipment is installed at each end host, and only end-to-end measurements are performed.

The computation of the previously detailed time-related QoS parameters requires the estimation for each packet of the time they spend in the network. Consequently, it is necessary to timestamp packets when they are transmitted and when they are received. This timestamping operation is typically done by packet capture mechanisms to happen as close as possible to the actual transmission and receiving dates in the Network Interface Card (NIC). Indeed, if timestamping happens in the host's application layer, the measurements include the time

Model	Bandwidth	Synchronisation	Clock precision	Price category
DAG cards - Endace [1]	10 Gbps	GPS or CDMA	7.5 ns	B
COMBOI-10G4TXT - Liberouter [2]	10 Gbps	GPS	not known	C
OmniAdapterTM 10G - WildPackets [3]	10 Gbps	GPS or CDMA	7.5 ns	C
TurboCap - Riverbed [4]	1 Gbps	software	1 μs	B
GigaStor 10Gb Wire Speed - Network Instruments [5]	10 Gbps	GPS	150 ns	C
IS4320E Packet Capture Card - Sysmate [6]	10 Gbps	GPS	100 ns	not known

TABLE I
COMPARISON OF THE HARDWARE-BASED PACKET CAPTURE SYSTEMS.

that packets have spent waiting to be processed by the host and not just the time that they have spent within the network. The first challenge is thus to have at our disposal efficient capture packet mechanism 1) to capture packets as close as possible to the transmitting and receiving times, and 2) to timestamp packets without incurring delay during this processing.

With a packet capture system at each end of its route, the packet is timestamped at the source and at the destination. But, these two devices do not have the same clock (i.e. the entity telling the time on the device): each device has its own clock in communication networks. Therefore, it is essential to synchronise the source's clock and the destination's clock to obtain accurate one-way measurements as they are computed by making the difference between two absolute clock reading values.

In the following, we firstly detail the available solutions for network packet capture. Subsequently, we present the issue of clock synchronisation in networks and the applicable solutions to obtain accurate timestamps. One should notice that timestamp precision and timestamp accuracy are strictly different although the accuracy cannot exceed the granularity. The precision refers to the granularity provided by the clock (i.e. time unit in which the time is provided), while the accuracy is determined in comparison with an absolute reference clock such as an atomic clock for instance (i.e. it represents the maximum difference between the clock and the absolute time).

III. PACKET CAPTURE SYSTEMS

Among the proposed solutions found in the literature to monitor network infrastructures, we can distinguish two main categories: the hardware solutions relying on dedicated hardware often linked to GPS (Global Positioning System) antennas and the software solutions relying on standard Network Interface Cards (NICs) and appropriate dedicated software.

The most emblematic solutions of these two categories are on one side the solutions based on Endace DAG (Data Acquisition and Generation) cards with GPS, and the solutions based on standard NICs with NTP (Network Time Protocol) and additional packet capture software on the other side. Figure 1 illustrates these different approaches and their issues. DAG cards can reach a 10 ns accuracy [1], whereas NTP-based solutions only provide millisecond accuracy [7].

While hardware solutions provide a high accuracy, they are expensive and often require GPS antennas and thus, cannot be easily deployed. On the other hand, software solutions present

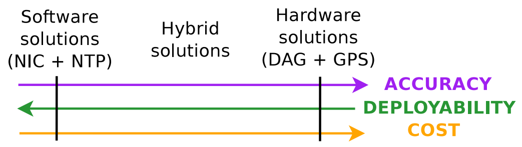
Software solutions (NIC + NTP)
Hybrid solutions
Hardware solutions (DAG + GPS)
ACCURACY
DEPLOYABILITY
COST

Fig. 1. Different available solutions for packet capture

a poor accuracy, but they are cheaper and more easily deployable as they do not require particular hardware. The literature provides also some hybrid solutions based on programmable NICs and special firmware for network measurements. The synchronisation of these cards can then be based either on GPS or on network synchronisation protocols such as NTP.

A. Hardware solutions

Table I summarizes the dedicated hardware currently available in the market and the main characteristics of each card: supported bandwidth, synchronisation method, internal clock precision and price category[1].

The hardware-based solutions massively rely on GPS synchronisation. This enables them to achieve a great accuracy for their internal clock, typically bellow the micro-second. As mentioned earlier, these dedicated solutions are in majority accurate but expensive, which is a major drawback in the design of a scalable solution for network metrology. Moreover, contrary to what is written in specification sheets, some of these dedicated cards do not support full-speed transfers and suffer from packet losses at high rates [8].

B. Software solutions

In contrast with the expensive and proprietary hardware-based solutions, numerous software-based solutions have been implemented and some of them are open-source. Some of the most recent and common ones are presented in Table II. These solutions make use of commodity network cards (NICs) and packets are generally captured at the user-level. The high number of such solutions shows that network metrology is a matter of concern for numerous applications, and that hardware-based solutions are not suitable in every case.

The numerous software-based solutions find their legitimacy on the high cost and scalability issues of hardware-based systems. Yet, these software solutions also suffer from severe drawbacks such as high CPU usage, low accuracy and packet

[1]Price categories are defined as follows. A: from 0 to 999 USD; B: from 1,000 to 9,999 USD; C: from 10,000 to 100,000 USD.

losses at high speeds. It would have been interesting to compare the CPU usage and packet losses for each solution. Yet, each paper of the literature provides results based on different packet sizes and different traffic rates [9], [10], [8].

Name	Timestamp precision	Licence
Libpcap - tcpdump [11]	μs	BSD
PF_RING [12]	μs	GNU GPL
nCap [10]	μs	proprietary
Libpcap-based solution [13]	ns	not known
Zero-p [14]	μs	not known
DiCAP [9]	μs	not known
DashCap [15]	μs	not known
Libtrace [16]	ns	free

TABLE II
COMPARISON OF THE SOFTWARE-BASED PACKET CAPTURE SYSTEMS.

Most of the software-based solutions rely on the *libpcap* library which provides a timestamp format with microsecond resolution. So, the timestamping accuracy of these solutions cannot go beyond this limit. Moreover, they all use the host computer clock to timestamp the packets, so the timestamping accuracy depends also on the computer clock's accuracy which often relies on NTP (Network Time Protocol) for its synchronisation. NTP achieves a millisecond accuracy [17] only. Another inaccuracy source comes from the fact that packets are treated in user-space and transmission delays from the NIC and from the kernel-space are highly unpredictable [10]. Some of these implementations have been developed for the purpose of research projects and are not available on the market.

C. Hybrid solutions

Hybrid solutions try to get the best of both worlds: accuracy and high performance from hardware-based solutions by using programmable network equipment which will be responsible for the packet capture; and deployability and low cost from software-based solutions that will be used to develop drivers for these network cards. Some of these solutions are presented in Table III.

The *ANME* box has been developed in the context of the OneLab initiative and is based on a VHDL-programmable ARGOS card. This card is linked to a GPS receiver to obtain a high clock accuracy. The Myricom solution is based on *Myri-10G* network adapters and on the *Sniffer10G* firmware which is installed on these cards and responsible for packet capture. The Intel solution relies on the Precision Time Protocol (PTP) and thus, requires to be linked to a PTP master to get synchronisation.

The *ANME* box presents a better clock precision, but this precision comes from the usage of a GPS receiver which restricts the scalability and deployability of this solution. On the other hand, the Myricom and Intel cards offer a lower clock precision which can be an issue for high-speed timestamping.

In fact, any NIC can be used to perform network metrology as demonstrated with the software-based solutions. Yet, non-specialized NICs require software implementations to handle packet capture and they offer a lower timestamping accuracy. Programmable devices can present an interesting compromise between accuracy and scalability. The hybrid solutions presented here rely on such programmable devices and come with firmware performing packet capture. Yet, the solutions which are not based on GPS synchronisation still lack accuracy to deal with high speed traffic.

IV. CLOCK SYNCHRONISATION METHODS

Timestamping for metrology purpose is a challenging issue that often requires synchronisation between the measurement devices. Usually, the devices are synchronised with *reference clocks* (e.g. GPS receiver, atomic clocks), and thus, the accuracy of the device's clocks depends on their precision, the precision of the reference clocks and the synchronisation method (evaluation of the latency). This synchronisation is complicated by the latency's asymmetry between any two network devices which is mainly caused by network congestion [21].

A software clock is based on a reference clock (which can be distant) and a local hardware counter which relies on an oscillator. The reference clock is accessed periodically through time servers using dedicated protocols like NTP or PTP for instance. The period of these communications is called the *polling interval*. The synchronisation to the reference clock is used to correct the natural drift of oscillators. The local hardware counter counts the 'ticks' of the oscillator at a given frequency. To describe it simply, the clock produces timestamps by converting the counter into seconds and adding a constant to set the time origin.

In the following, we present the most commonly used methods for network synchronisation in distributed systems. Other less generic solutions have been proposed to measure network latency without synchronisation like sending pairs of probe packets [22].

A. Global Positioning System (GPS)

The Global Positioning System comprises 24 satellites, and each of them has an atomic clock providing accurate time. On Earth, the time signals transmitted by the GPS satellites are collected by GPS receivers. The receivers then provide a one pulse-per-second output signal with accuracy on the order of few tens of nanoseconds for the more accurate receivers [23].

The GPS receivers can be used for clock synchronisation but they come along with high costs and efforts. Indeed, they require antennas to be mounted outside and with a large view angle to the sky (seeing at least 4 GPS satellites), while measurement devices are close to the infrastructure often located in server rooms or at the basement. These two issues prevent GPS solutions (i.e. one GPS per network device) to be scalable. Yet, GPS receivers can be used within synchronisation infrastructures as reference clocks.

B. Network Time Protocol (NTP)

The Network Time Protocol (NTP) is one of the oldest still in-use Internet protocols. Thus, NTP implementations are commonly available in most of the operating systems (OS).

Name	Bandwidth	Network card	Synchronisation	Clock precision	Price category
ANME box - OneLab [18]	not known	ARGOS FPGA	GPS	10 ns	B
Sniffer10G firmware - Myricom [19]	10 Gbps	Myri-10G card	computer clock	500 ns	A
Intel Ethernet Controller I350 Driver [20]	1 Gbps	Intel I350 Card	PTP	100 ns	A

TABLE III
COMPARISON OF THE HYBRID PACKET CAPTURE SYSTEMS.

NTP is a distributed network clock synchronisation protocol. It uses a hierarchical system where each level is called a *stratum*. At the top, the Stratum-0 is constituted with reference clocks such as GPS receivers, atomic clocks or radio clocks. These reference clocks are directly linked to Stratum 1 NTP servers. These Stratum 1 servers, also called time servers, are able to answer the timing requests of Stratum 2 servers. At each level, the servers discuss with several servers of the upper level, and they also peer with other servers from the same level to improve their clock accuracy [7]. For Unix-based systems, the NTP clients use a daemon, called *NTPd*, which runs continuously in user space. However, the system clock (synchronised by using NTPd) is implemented in kernel space to decrease the access latencies to the time values.

The synchronisation mechanism between a client and a server is described in Figure 2. At t_o (origin timestamp), the client sends a packet to the server. The server receives the packet at t'_r (receive timestamp). The server then sends a message back to the client at t'_t (transmit timestamp). The client receives this message at t_d (destination timestamp). The problem with these timestamps is that they are not produced by the same clock and so, they do not have the same time frame of reference. So, the client NTPd should compute the difference between the two clock, which is called the *offset*, in order to correct it.

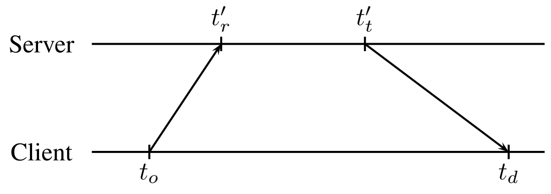

Fig. 2. Synchronisation mechanism between a client and a server

If the one-way delays between the client and the server on one side and the server and the client on the other side are perfectly identical (symmetrical delays), then the offset is defined as:

$$offset = \frac{(t'_r - t_o) + (t'_t - t_d)}{2} \tag{1}$$

If the client and the server are perfectly synchronised, the offset is null. NTP uses Formula (1) to compute the offset, thus assuming that the one-way delays are symmetrical and that clock time varies only linearly during the exchange. The computed offset is then used as a correction to adjust the system clock as shown on Figure 3.

The system clock takes as input a hardware counter and the correction parameters provided by NTPd. To timestamp a

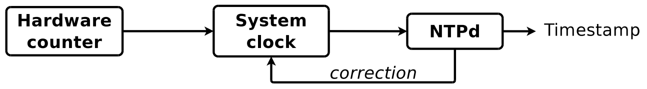

Fig. 3. NTPd feedback loop

packet, NTPd uses the system clock as described in Figure 4. When the packet has crossed all the protocol stack (in kernel space), it is passed through a socket to the user space and only then, it can be timestamped by NTPd.

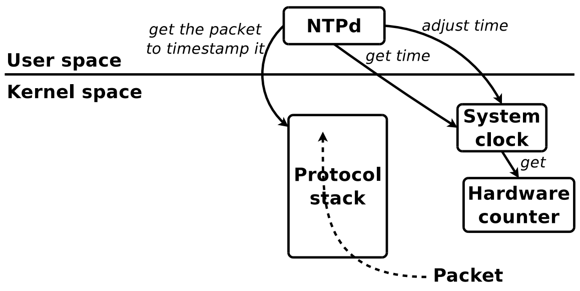

Fig. 4. Timestamping process with NTPd

The main drawbacks of NTP are its poor accuracy and stability that makes it unsuitable for high-performance networks. Clock synchronisation convergence can take many hours using NTP, and NTPd frequency adjustments can cause inconsistencies in timestamps [24].

C. Precision Time Protocol (PTP)

The Precision Time Protocol (PTP) aims at synchronising clocks over a local area network (LAN). Each networking equipment needs specific hardware to support this protocol. It especially targets local systems which require an accuracy higher than the one provided by NTP, but cannot afford the cost of GPS receivers for every nodes.

PTP has been first standardized in 2002 under the denomination *IEEE 1588-2002*. This synchronisation protocol is based on a master-slave model. Each slave runs the best master clock algorithm (BMC) to find its most suitable master. The offset calculation is then computed by using the same formula as NTPd (Formula (1)) and thus, with the same assumptions. On LANs, PTP can achieve an accuracy in the sub-microsecond range [25]. Yet, it is less robust to deviations than RADclock [26].

D. Robust Absolute and Difference Clock (RADclock)

The Robust Absolute and Difference Clock (RADclock) [27] is a system for network timing providing a difference clock to measure accurately the time elapsed between two events, and an absolute clock to get timestamps (like in the current system clocks). The timestamping process of RADclock is performed in the kernel. Unlike NTPd, the correction parameters are not directly applied to the system clock. They are instead used only when a timestamp is required, thus limiting the impact of frequent adjustments and inconsistencies.

Several studies have shown the great robustness and accuracy of RADclock compared to other solutions like NTP or PTP [27], [26], [28], [29]. Yet, it relies ultimately on NTP Stratum-1 servers, which are the reference clocks for this method.

Method	Targeted networks	Required installation
GPS	Small number of nodes	Antennas directed to satellites
NTP	All (not requiring too much accuracy)	NTP Stratum-1 servers (reachable from anywhere over Internet)
PTP	Local Area Networks	PTP cards and PTP masters
RADclock	All	RADclock patch and NTP Stratum-1 servers

TABLE IV
SUMMARY OF THE AVAILABLE SYNCHRONISATION METHODS

Table IV summarizes the available synchronisation protocols with their required equipment. All methods require an access to reference clocks located either on satellites, NTP servers or PTP masters.

V. A DEEPER LOOK AT NIC-BASED TIMESTAMPING

As explained before, current hardware and software solutions for packet timestamping present major drawbacks. Hybrid solutions using NIC-based timestamping could be a good trade-off between accuracy and cost. Yet, the non-GPS-based solutions, presented in Section III-C, lacks of accuracy. This section presents some ideas and comments towards an effective capture/timestamp hybrid system. Software-based metrology solutions (i.e. using standard NICs) rely on the operating system kernel to timestamp the arriving and departing packets. This process is by definition inaccurate since these timestamps do not reflect the actual arriving or departing time of the packets. In particular, they depend on the time required by the kernel to process the packets and to access the clock.

As shown in Figure 5, in the case of a packet timestamping processed in the kernel, the packet has already crossed the NIC by entering through the port and going out through the NIC bus to reach the kernel space. In the case of a timestamping in the user space (application space), the process suffers from even more latencies since it has to use system calls to access the values of the kernel clock, and the packet has to cross the whole protocol stack in the kernel. Another solution consists of timestamping packets in the NIC by using the host's clock (which can have a higher resolution than the NIC's clock). But this option has to deal with consequent latencies induced by the NIC's bus used for communication between the host and the NIC. Reading the NIC time can take over $1 \mu s$ which is significant for high-speed traffic, and this delay is variable [30].

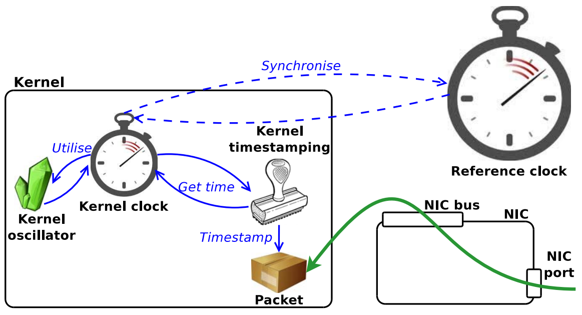

Fig. 5. How packets are timestamped in the OS kernel

To obtain a timestamp close to the reality and get rid of host-NIC communication latencies, packets should be timestamped by the NIC in a similar way as PTP packets in PTP-enabled NICs. In the case of a timestamping in the NIC, the inaccuracy of timestamps comes from several major reasons as illustrated in Figure 6:

1) the method used to synchronise the clock with an external reference clock;
2) the precision of the internal clock itself;
3) the method employed to access the internal clock when a timestamp is required (e.g. system call latency);
4) the method used to timestamp the packets (individually upon arrival or periodically in batch).

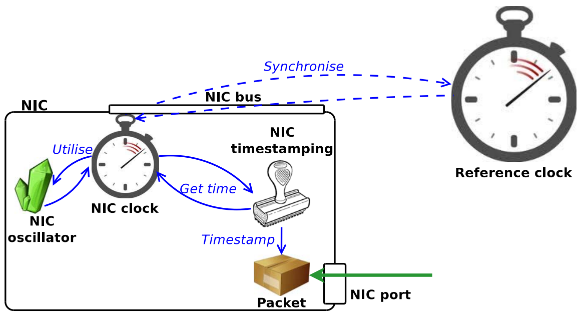

Fig. 6. How packets are timestamped in the NIC

The reference clock can be a GPS receiver, the host's clock or a distant clock server (e.g. NTP server or PTP master). In the case of a synchronisation of the NIC's clock with the host clock, it is required that messages are passed through the NIC's interface bus (e.g. PCI bus). This process causes latencies on the order of 1 μs for a PCI bus[2]. For this reason, it seems mandatory to timestamp packets in the NIC using the NIC clock directly and not the host clock. However, NIC-host communications are nevertheless required to keep both clocks synchronised and to be able to provide timestamps that will be usable in the host (i.e. converted in host time).

[2]http://www.myricom.com/kb/index.php?title=What_is_the_timestamp_accuracy_in_Sniffer10G?

Another major issue comes from the packet treatment process of the NIC. Even if the NIC timestamps the packets, they may not be timestamped upon arrival. This operation can be dispatch-driven to deal with high-speed performance concerns (reducing packet losses) [30], and is subject to latencies (up to 1μs for a Myri-10G card for example according to the specifications provided by Myricom[2]). As a result, several packets may obtain the same timestamp even if they did not arrive at the same time in the NIC and even if the timestamping precision (granularity) was high enough to provide them with different timestamps.

VI. Conclusion and future work

As detailed in this paper, various solutions exist to monitor the quality of service of high-speed networks. However, most of the available solutions are either expensive and not scalable but accurate and efficient, or inaccurate and presenting poor performance but easily deployable. Thus, designing a network measurement infrastructure which is accurate, inexpensive and easy to deploy in comparison with existing solutions appears as an unsolved challenge. Such an instrumentation platform could benefit many users and applications. For instance, the PetaFlow application [31], which is a distributed, interactive and high-performance application, illustrates this instrumentation requirement by its highly time-sensitive nature.

Our future work aims at taking advantage of RADclock and cheap network cards able to perform hardware timestamping to propose an inexpensive and accurate metrology platform. The major issue in designing such an instrumentation system with these components is to deal with two different clocks (i.e. host and NIC clocks) which are a priori not synchronised as they are based on different oscillators. To validate this new metrology platform, we will use the PetaFlow test-bed which comprises a 1 Gbps dedicated trans-continental link between France and Japan, and a state-of-the-art end-to-end synchronised capture system using DAG cards.

Acknowledgment

The authors thank the partial support of the joint ANR-JST project PetaFlow (CSD1-ANR-09-BLAN-0376-01) and of the Inria Explorateur Grant awarded to Anne-Cécile Orgerie to visit the University of Melbourne.

References

[1] DAG cards, Endace corporation. [Online]. Available: http://www.endace.com/endace-dag-high-speed-packet-capture-cards.html

[2] COMBOI-10G4TXT, Liberouter. [Online]. Available: http://www.liberouter.org/card_comboi_10g4txt.php

[3] OmniAdapterTM 10G, WildPackets. [Online]. Available: http://www.wildpackets.com/products/network_recorders/omniadapter_analysis_cards/omniadapters_10g

[4] TurboCap, Riverbed. [Online]. Available: http://www.riverbed.com/us/products/cascade/wireshark_enhancements/turbocap.php

[5] GigaStorTM 10 Gb Wire Speed, Network Instruments. [Online]. Available: http://www.netinst.com/products/observer/gigabit.html

[6] IS4320E- Network 10 Gbps Ethernet Packet Capture Card, SYSMATE CO LTD. [Online]. Available: http://www.sysmate.com/new2/eng/pro_is4010.html

[7] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482 –1493, Oct 1991.

[8] S. Ubik, "Gigabit ethernet packet capture tools," CESNET, Tech. Rep., Sept. 2005. [Online]. Available: http://luca.ntop.org/ncap-evaluation.pdf

[9] C. Morariu and B. Stiller, "DiCAP: Distributed Packet Capturing architecture for high-speed network links," in *IEEE Conference on Local Computer Networks (LCN)*, Oct. 2008, pp. 168 –175.

[10] L. Deri, "nCap: wire-speed packet capture and transmission," in *Workshop on End-to-End Monitoring Techniques and Services*, May 2005, pp. 47 – 55.

[11] tcpdump/libpcap. [Online]. Available: http://www.tcpdump.org/

[12] PF_RING, ntop. [Online]. Available: http://www.ntop.org/products/pf_ring/

[13] P. Orosz and T. Skopko, "Software-Based Packet Capturing with High Precision Timestamping for Linux," in *International Conference on Systems and Networks Communications (ICSNC)*, 2010, pp. 381 –386.

[14] X. Xu and Z. Li, "High-Speed Packet Capture Mechanism Based on Zero-Copy in Linux," in *International Conference on Biomedical Engineering and Informatics (BMEI)*, Oct. 2009, pp. 1 –5.

[15] M. Dashtbozorgi and M. Azgomi, "A high-performance software solution for packet capture and transmission," in *IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, Aug. 2009, pp. 407 –411.

[16] S. Alcock, P. Lorier, and R. Nelson, "Libtrace: A trace capture and processing library," University of Waikato, Hamilton, New Zealand, Tech. Rep., May 2010.

[17] L. De Vito, S. Rapuano, and L. Tomaciello, "One-Way Delay Measurement: State of the Art," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 12, pp. 2742 –2750, Dec. 2008.

[18] ANME (Advanced network monitoring equipment) box, OneLab. [Online]. Available: http://www.onelab.eu/index.php/services/testbed-components/anme-box.html

[19] Myricom Sniffer10G. [Online]. Available: http://www.myricom.com/support/downloads/sniffer.html

[20] Intel Ethernet Controller I350. [Online]. Available: http://www.intel.com/Assets/PDF/prodbrief/I350_Family_Product_Brief_v001.pdf

[21] N. Simanic, R. Exel, P. Loschmidt, T. Bigler, and N. Kero, "Compensation of asymmetrical latency for ethernet clock synchronization," in *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, 2011, pp. 19 –24.

[22] G. Wei-Xuan and Y. Shun-Zheng, "Inference of One-Way Queuing Delay Distribution Using Packet-Pair Probes without Clock Synchronization," in *IFIP International Conference on Network and Parallel Computing Workshops (NPC Workshops)*, Sept. 2007, pp. 169 –175.

[23] J. Levine, "Introduction to time and frequency metrology," *Review of Scientific Instruments*, vol. 70, no. 6, pp. 2567 –2596, June 1999.

[24] A. Harrison and P. Newman, "TICSync: Knowing when things happened," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 356 –363.

[25] R. Schmidt and B. Fonville, "An NTP Stratum-One Server Farm Fed By IEEE-1588," in *Annual Precise Time and Time Interval (PTTI) Meeting*, 2010, pp. 111–126.

[26] J. Ridoux and D. Veitch, "The Cost of Variability," in *International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS'08)*, Sep. 2008, pp. 29–32.

[27] D. Veitch, J. Ridoux, and S. Korada, "Robust Synchronization of Absolute and Difference Clocks Over Networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 417 –430, april 2009.

[28] J. Ridoux and D. Veitch, "Ten Microseconds Over LAN, for Free (Extended)," *IEEE Transactions on Instrumentation and Measurement (TIM)*, vol. 58, no. 6, pp. 1841–1848, June 2009.

[29] J. Ridoux, D. Veitch, and T. Broomhead, "The Case for Feed-Forward Clock Synchronization," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–12, 2011.

[30] P. Ohly, D. Lombard, and K. Stanton, "Hardware assisted precision time protocol. Design and case study," in *LCI International Conference on High-Performance Clustered Computing*, 2008.

[31] P. Gonçalves, X. Grandchamp, X. Pelorson, B. Raffin, A. Van Hirtum, P. Vicat-Blanc, K.-i. Baba, J. Cisonni, Y. Ebara, K. Nozaki, H. Ohsaki, S. Wada, T. Kawamura, K. Koyamada, E. Sakane, N. Sakamoto, and S. Shimojo, "Petaflow - A Project Towards Information and Communication Technologies in Society," in *IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, 2010, pp. 347–350.