Clock Synchronisation in Industry-standard Computer Hardware

Markus Peuhkuri
Networking Laboratory
TKK Helsinki University of Technology
Espoo, Finland
Email:markus.peuhkuri@tkk.fi

Antti Gröhn
Nokia Siemens Networks
Espoo, Finland
Email:antti.grohn@nsn.com

Oskari Simola
Networking Laboratory
TKK Helsinki University of Technology
Espoo, Finland
Email:oskari@netlab.tkk.fi

***Abstract*—Accurate timing is an essential component for various areas of experimental research. Although expensive capture cards provide accurate time stamps for packets seen in networks, those do not help in synchronising applications or in-kernel processes. We present a moderate-cost approach to provide sub-microsecond accuracy on common PC hardware using custom-built clock card. The synchronisation accuracy is improved several orders of magnitude compared with synchronisation over network and an order of magnitude compared with pulse-per-second signal on a serial port. Clock accuracy does not significantly suffer from a high I/O load on the computer.**

I. Introduction

Experimental research using computers often needs to keep record of time. For some experiments, quite a coarse accuracy is enough and synchronisation among devices is not required. Nevertheless, often the lack of inter-system synchronisation or even inaccuracies in time keeping within one system, may cause significant problems in analysing and interpreting results.

Experimental research is most often conducted by using commercial of the shelf products namely standard PC hardware. This is simply due to scale of economy - large production quantities lower the unit prices. Aside from this PC-compatible hardware is prone to weaknesses that often set limits to the research and lower the quality of results. Accuracy and synchronisation of time are some of these issues.

On-board crystal oscillators (XO) provide base of time synchronisation and accuracy of time keeping in PC-compatible hardware. On short timescales ($< 1\,\mathrm{s}$) the stability of the crystal oscillator is usually adequate, but the long-term stability is often not [1]. Crystal oscillator frequency depends on temperature. In PC hardware crystal is mounted to the motherboard where it is directly influenced by heat dissipation of internal component and cooling by fans. This can cause high variation into the crystal temperature and therefore variation into the oscillation frequency.

Sub-second accuracy in synchronising computers in networked environment is easily achieved using NTP or SNTP protocols [2], [3] and time-keeping becomes a non-issue in day-to-day operations. For applications targeting millisecond or better accuracy that is typical and frequently required in networking research, this is not sufficient. Capturing of packets from a gigabit Ethernet link requires clock resolution of better than microsecond and scheduling of packets in deadline based packet scheduling this figure is order of magnitude lower. Active measurements over LAN or even WAN typically require minimium clock accuracy of few hundred microseconds to find differences among test instances.

A. Small Deterministic Systems

It is possible to build systems that are deterministic and can provide accurate timing of events. Whereas systems expand, this becomes more difficult. One of the biggest challenges in using general purpose hardware — a common PC — is to achieve reasonable accuracy in measurements without having to resort to task-specific and expensive hardware.

To get accurate time-stamps for incoming packets, for example, the hardware design must support immediate time-stamping. Custom-built equipment where both time-keeping and packet capture are embedded onto a single card provides this. [4]. Aside from this, it is not a cost effective approach when a large scale deployment or variable requirements are considered. Accurate system level timing is more often preferable solution, as the target measure may not be only packet level operations of network rather in processing of application information elements or execution of internal algorithm. Packet level time stamping does not help on these issues, how accurate it is.

B. Time Synchronisation for Network Research

We have designed our synchronisation system for networking research in our mind, especially for synchronisation of different traffic generation and analysis platforms. Our networking test bed is based on mixture of commercial and experimental network devices. The research focuses on network based experimentation of different networking algorithms and protocols. For those experiments synchronisation of different traffic generation and analysis points is extremely important. To be able to reproduce traffic patterns and events for multiple occasion and to be able to analyse differences among algorithms requires microsecond resolution.

For example, our adaptive delay based scheduling algorithm calculates short and long range queueing delays for the packets and makes scheduling decisions based on delay indices. To analyse design choices we must replay traffic patterns over and

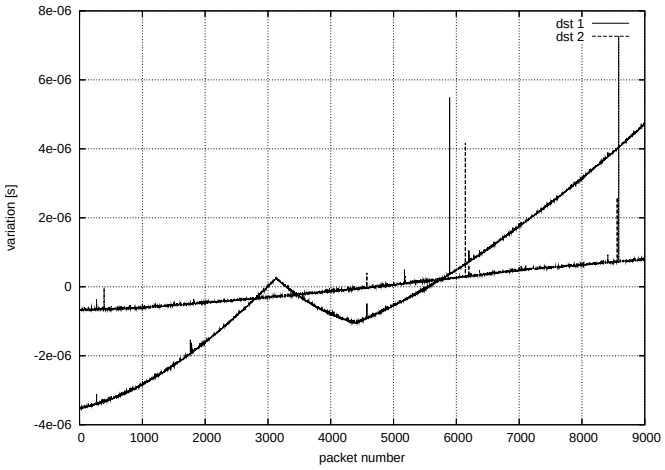

Fig. 1. Individual packet delay compared with mean delay on two receiver with unsynchronised clocks

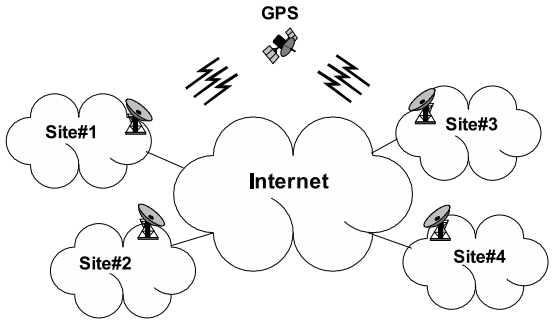

Fig. 2. Inter-site synchronisation using GPS satellite

over again with the resolution of actual scheduling events e.g. microseconds. Without good time-synchronisation among load generators and analysers we would not be able to distinguish meaningful differences on the operation of algorithm from the noise generated by the clock error among devices. Accurate timing in event logs makes analysing measurements easier.

An example of problems caused by loosely synchronised clocks can be seen from Fig 1. It shows how measured multicast packet delays evolves over half of hour on two receivers. A significant skew in delays and also changes in delay slope can be seen even if network state has been quite stable. From that data, it is difficult to automaticly determine what was the actual delay at each point of time.

Our system requires only one GPS receiver to feed the time information to all of our devices. One can argue that the wall clock time received from the GPS is not important for our research and with a high quality oscillator we shall achieve the same. This is true if we conduct all of our research on the close vicinity of our time source. Notwithstanding, once we have distributed research platform we need inter-site synchronisation which can be achieved easiest with GPS information, as in Fig 2. This inter-site operation makes it possible to build large scale virtual laboratories with high precession time synchronisation of devices located close and far-away. We would use these virtual laboratories basically for the same research we do in our closed environment with the addition to be able to run multi-site traffic analysis like active probing or analyse media distortion when we have accurate time reference.

The structure of our system comes even more relevant when we start speaking of commissioning new networks. In general, high dependability networks are built within electromagnetically shielded rooms located below ground, places with poor GPS coverage. All information passing in and out from these rooms is delivered in optical format. Our system has this conversion naturally built in and also makes possible to deliver this information in a daisy-chain manner e.g. external distributor and internal distributor.

We start our work with a review on how time is managed in PC computers[1] and how time can be synchronised using protocols. Then we introduce our approach to provide accurate synchronisation in Sect. III-B and compare it with other approaches in Sect. IV. Finally, we conclude our work on Sect. V and propose for further work.

We assume that the reader is familiar with synchronisation of digital clocks. Good sources for information are [1], [5], [6] and NTP documentation at `http://www.ntp.org`.

II. Time Keeping on PC Computers

A typical PC computer motherboard has several clock circuits embedded onto it. Some of these originate from the first IBM PC, other are later, improved additions. Each computer has a battery-backed real time clock (RTC) that keeps wall clock time even if the computer is without power. This clock provides time for the operating system at boot-up, but because the RTC has only resolution of one second, a computer cannot keep time with better accuracy when it is powered off.

Original PC had a timer chip, 8254, that provided a timer interrupt when it counted to zero from a preset value. This function is typically embedded into south bridge chip on modern motherboards, but the function is still the same: there is a 14.31818 MHz XO and the timer counts down from a set value. An operating system sets a suitable value into a register to receive right number of interrupts per second. Typical values for the tick rate are 100, 250, or 1000 interrupts per second.

When the operating system needs to know current time, it checks the last time stamp value from previous timer interrupt and interpolates present time based on the current value in the 8254 register. Time stamp counter (TSC) provides high-resolution time stamp with a single processor instruction on modern processors. The TSC is incremented in every clock

[1]Here, the focus is on IA32 architecture. We use Linux as an operating system in our presentation, but other operating systems should have similar mechanisms.

cycle of the processor, thus providing a 0.5 ns resolution for a 2 GHz processor.

Contemporary motherboards provide also other timers, for instance high precession event timer (HPET) and timers supporting power-saving features essential for laptops. An operating system may support several of these timer methods and select the best one at the boot time. Nonetheless, two basic weakness are in every one of the timer methods.

The first problem is that all timers rely on oscillator running on the motherboard. Although a crystal oscillator (XO) provides good short-term stability, its frequency depends on temperature and other environmental factors [7]. Typical XOs found on PC computers are not temperature compensated and their frequency is not accurate as that won't bring any significant advantage on normal operation of computer.

The other problem is that there is no method to feed an external time impulse, like a seconds epoach, directly to this circuit. The only method is to use CPU to detect signal transition and then record current time value. Even if this may be feasible at system bootup, using similar method over normal operation will slow down the system.

Still another problem is that the system may not be able to react on the interrupt fast enough. Even if the timer interrupt is the highest in priority, some operation that disables interrupts may cause extra delay resulting in clock slowdown. The HPET performs better in this respect as it stores counter value at point of interrupt.

A. Time Synchronisation over Network

As computers are interconnected with ever faster networks, it is convenient to keep computers in correct time using network messages.

Several protocols are developed for that purpose, starting from a simple daytime [8] and time [9] services that provides clock resolution of one second. To achieve a better accuracy, there exists two protocols: Network Time Protocol (NTP) that is appropriate to use over Internet, and IEEE 1588 Precision Time Protocol that is appropriate to use among directly connected devices over local area networks. Standardised ways are developed to distribute timing information over dedicated links, one of those are IRTG time codes.

1) Network Time Protocol: Time synchronisation using NTP protocol is based on exchange of time stamps over network and then estimating the correct time based on those time stamps. The NTP servers form a tree structure, where a root servers (stratum 1) are synchronised to reference clocks. Multiple NTP root servers that synchronise typically to a GPS clock or to a atom clock are in Internet. Current list[2] lists 150 stratum 1 servers. About half of those provide open access.

When a computer is synchronised over the network using NTP protocol, it is typical to achieve few millisecond accuracy in a low-delay network. The synchronisation accuracy greatly depends on network stability and on how many servers there are in the chain from the root time server (i.e. stratum number).

[2]http://ntp.isc.org/bin/view/Servers/StratumOneTimeServers

A reference clock can be embedded into the NTP server or then the NTP software is run on common computer hardware and interfaced to a reference clock. The reference clock typically provides a serial output for a time stamp and a pulse-per-second input to signal exact change of second [10]. When using the time stamp from the serial line, the typical accuracy of time-keeping is approximately in the order of 10 ms.

The pulse-per-second (PPS) input is connected to a computer, typically to a DCD pin on a serial port that triggers interrupt at state change. The interrupt is served and a correction factor is adjusted as needed. This improves accuracy significantly: on an unloaded computer the estimated error can be close to 1 μs. Further, if the machine is heavily I/O-loaded this accuracy is reduced because of increased interrupt contention. As a result, one must keep processing load low and this is the common reason many stratum 1 servers provide only limited access.

2) IEEE 1588 PTP: The NTP protocol is designed for wide-area networks and thus deploys complex algorithms to filter out delay variations resulting from queueing delays in routers. Although if the network path has only few devices, it usually has more deterministic delays. If we have two devices that are directly connected, the delay is the serialisation delay of the packet. Also an Ethernet network having only hubs and repeaters will have constant one bit time delay without collisions.

The delays increase because of queueing in switches, specially store-and-forward ones. Even though Ethernet networks provide priorities (IEEE 802.11p) there may be also other traffic on the highest priority. The lower priority packets can also cause delays as packet transmissions are not preempted. The maximum delay is one or two maximum frame length transmission delays (122 μs for 100 Mbit/s networks and 74 μs for gigabit speeds if jumbo frames are used) per switching device in path.

The design of IEEE 1588 protocol is simple so it can be implemented on hardware level [11]. The clock node sends periodically two messages: an synchronisation message and followup message indicating the actual time the synchronisation message was sent. In addition, the clients may send delay estimation messages where symmetrical delays are assumed. It would be optimal to have all networking hardware to support IEEE 1588, but none of major switch vendors has products supporting it.

3) IRIG Serial Time Code: There exists protocols to transmit current time code using electrical or optical transmission. One of those is family of IRIG serial time codes issued by Inter Range Instrumentation Group of the Range Commanders Council under the authority of the US Department of Defense. The same signal transmits both wall clock time and provides accurate timing information using frame structure, bit timing and carrier signal. Multiple different formats with different bit rates are defined and the most common one is Format B that sends 100 pulses per second. It may be transmitted either with width coded signal or with amplitude modulated sine signal at 1 kHz or 1 MHz [12].

B. Uncertainty Sources in Time Measurements

Even if the operating system is assumed to know exactly the current time, the measurements made within the operating system's clock scope may not be as accurate. A simple example is described here: We want to capture and time stamp packets from an Ethernet network using a standard network interface card.

First, the network interface card (NIC) captures a packet and signals operating system with an interrupt. The operating system then reacts to the interrupt signal and retrieves the packet from the NIC provided that the PCI bus is available. If the bus is not available, the processor must wait it to become available. Now, if an another packet has arrived meanwhile, the processor is able to retrieve both packets at the same thus making it appear that the packets have arrived at the same time [13]. Many high-performance NICs can limit number of interrupts per second to improve throughput making this phenomenon worse.

Other factors in the operating system, often resulting from interrupt latency, also add up to timing inaccuracies in the PC environment. Therefore, it is necessary to carefully evaluate and pinpoint possible sources of inaccuracies in measurements. On the other hand, it is quite difficult to directly measure the latencies as the time perceived by the operating system is not directly visible on any signal lines where it could be measured and compared to an other time source. Even if some readily available I/O signals would be available, they are not suitable for sub-microsecond measurements in general. For example, changing pin state on a parallel port takes one microsecond making it too slow for that purpose.

III. Improving Synchronisation Accuracy

One approach to improve clock accuracy is to replace on-board crystal oscillators with ones having better stability. If the crystal oscillator is temperature compensated (TCXO) or kept in a stable temperature (oven-controlled XO, OCXO), stability of 10^{-8} or 10^{-11} on day-to-day basis can be achieved [1]. Replacing the on-board oscillator is not a trivial task and also needs motherboard-specific design and testing. Furthermore, the problem of synchronising clocks on multiple systems together or to UTC time would not be solved.

A. Using Pulse-Per-Second for Synchronisation

A simple and easily available approach would be distributing PPS signal to each computer using either serial or parallel port signals. It is possible to have seconds numbering over network with daytime or NTP protocol and use PPS to signal a precise seconds epoch. The benefit for this approach is its low cost. The construction is slightly more complicated if one wants to provide a galvanic isolation among computers, but still below 50 euros per unit.

Using serial and parallel for accurate synchronisation has problems regardless of them being readily available. First, the ports have protection circuits for electrostatic discharge (ESD) that limits the clock signal rise time. A more significant source for error is the fact that both serial and parallel port interrupts

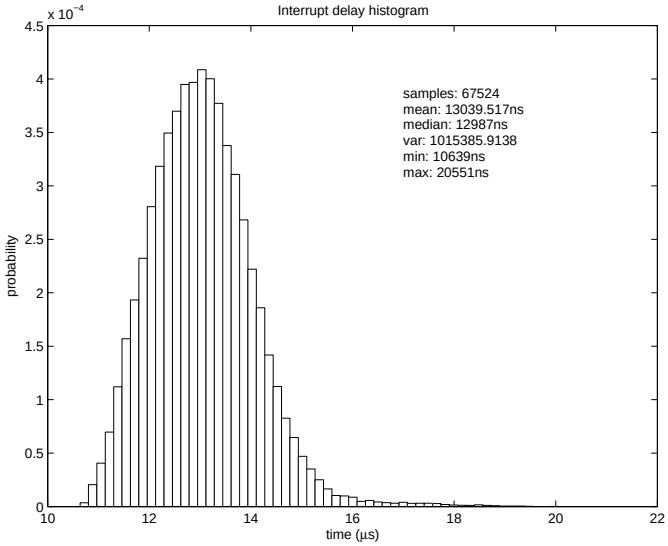

Fig. 3. Serial port interupt delay histogram on an unloaded computer

are low priority ones. This means that if some higher priority interrupt, like disk or network I/O, is running, the interrupt will be delayed until the higher priority one is served. Also, a high-priority interrupt may preempt a low-priority one causing error to time estimation.

The PPS interface for Linux operating system provides an echo function: as soon the PPS interrupt routine starts, it changes state of another pin on serial port resulting in a PPS echo signal. Using a timer card, it is possible to measure a time interval between PPS and PPS echo signal. Fig. 3 shows the delay between these two signals on PC described on Section IV.

Over the 18 hour measurement period, the echo delay was less than $16.7\,\mu$s for 99.5 % of time. What is more, when the computer was loaded with network and disk I/O load, the echo delay was higher than $19.8\,\mu$s for 0.5 % of time. The above measurement was done using RT-patched version of Linux kernel with serial interrupt set as highest possible priority. When a similar measurment was run with an non-realtime version, longest delays were significantly higher and and some PPS signals were left without echo altogether.

B. Providing Accurate Time Using Add-on Card

Multiple sources of error to disturb the accurate synchronisation of PC hardware, as we have seen. The largest problem is the uncertainty of undetermined latency caused by serving interrupt. If it is possible to reduce or remove this component, the accuracy would be significantly improved.

Again, changing priority of interrupts would require some hardware modifications and we consider this as an inappropriate solution for a wide use. Yet, we can change our assertion from "seconds epoch was just" to "current time is now". We will check time when it best suites us and when it can be done with minimal latency. Based on this information we make appropriate corrections to the timing information.

The main idea is to construct an add-on card, "SynPCI", that has two counter registers. The first one is a running counter that is reset each time it is read. The other counter records the value of the first counter when a PPS signal is received. The clock source for the first counter is a voltage-controlled oscillator (VCO) that is phase locked to incoming 10 MHz signal from a GPS clock.

Each time a timer interrupt is triggered, a 32-bit value is read from the first counter register. This includes 1-bit PPS indicator and 31-bit value from the first counter. The counter size is sufficient as it is read 100 or 1000 times a second. The kernel time-of-day value `xtime` is updated according to value read from the register.

The PPS indicator is then used to detect if a PPS pulse is signalling seconds epoch received between timer interrupts. If the bit is set, then the other register is read and an correction is applied to `xtime` if needed.

1) How is this better than PPS?: Compared with standard PPS-signal usage, the main improvement in this method is that the interrupt latency does not have an effect whatsoever. The time between interrupts may not exceed 2 s, but else they can take place whenever. If access to the PCI bus is delayed it has no effect to accuracy, since the counter value is latched only when it is read. The read process takes about 300 ns on 133 MHz PCI-X bus and 450 – 650 ns on 33 MHz PCI bus, depending on motherboard architecture

2) Design of the SynPCI card.: Figure 4 shows the main components of the SynPCI card. The main design consists of one Lattice LCMXO1200 complex programmable logic devices (CPLD). The same chip provides PCI bus functions and counter functions. The card takes two optical inputs, one for PPS signal and an another for 10 MHz clock signal. One can select VCO frequency as any multiple of 10 MHz upto 400 MHz with software. Selecting a right multiplier for PLL makes it possible to count with desired resolution at 2^N nanoseconds resulting faster execution.

Using optical signals has advantages: With optical signals one does not have to care about grounding issues and optical signals make it possible to transfer signal for longer distance than when using electric signals. Using optical transceivers and receivers was not significantly more expensive than to provide galvanic isolation on card.

The card is designed so that it is possible install that vertically on 1U rack chassis, provided that mechanical construction of chassis allows that. The card has LED indicators for incoming signal to help one to verify correct installation of fibre cables. A connector with 20 I/O signals can be used to distribute signals or receive inputs in future applications. The connector and free capacity on in-system programmable CPLD can be used for interfacing with other systems.

In addition to the SynPCI card, the system needs also an distribution board that takes electric PPS signal and 10 MHz signal from the GPS clock and outputs a desired number of optical signals. Using optical signals provides both galvanic isolation and extended range with reasonable low cost. The SynPCI card itself is about 100 euros per card, the distribution board is about 50 euros per card and a GPS receiver with 10 MHz output is about 1,500 euros. Thus for a single site with 20 synchronised PCs will have equipment cost of 4,500 euros, 225 per PC.

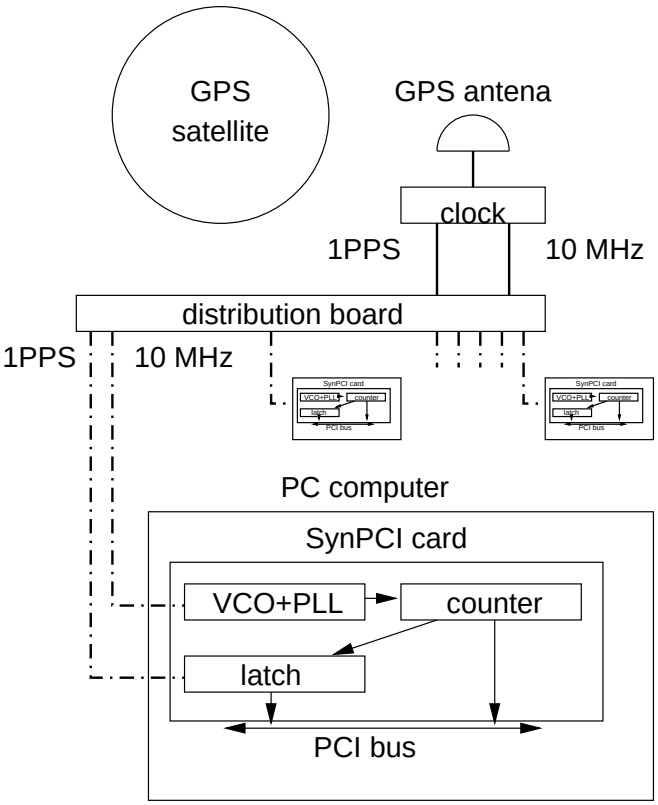

Fig. 4. Structure of SynPCI syncronisation system

3) Delay Analysis of SynPCI architecture: It is important to identify delay in the system and if some part of the delay is variable, and take that into account. We use Trimble Thunderbolt GPS disciplined clock to provide both 1 PPS signal and 10 MHz signal. According to manufacturer, it provides 20 ns accuracy for PPS signal. The device is connected with 50 Ω coaxial cable to distribution board. If we assume 2 m length of cables, that will result a 10 ns constant delay.

The conversion from electrical signal to optical and back to electrical signal results some delay. According to measurements, our implementation results 25 ns delay. Fibre optic cables result also some additional delay, about 5 ns per meter. If cables have significantly different length, it must be taken into account.

We can conclude that delays resulting from the signal conversion and distribution are constant, about 85 ns with 10 meter optical cables. This can be taken into account in syncronisation, if needed.

C. Related Work

A quite similar approach to build a card is presented in [14, p.63]. In comparision with our system, according to the description the card is more complex and thus may be more

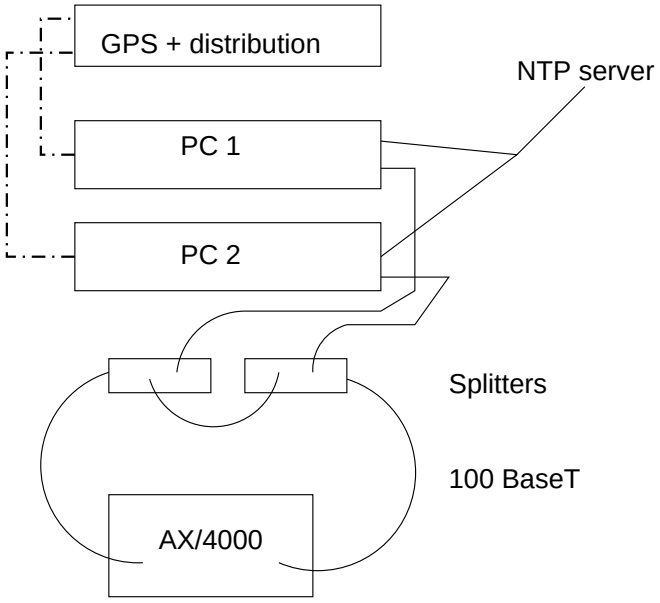

Fig. 5. Measurement setup

expensive. That is not an issue if only one or few cards are installed at a single site. Our plan is to deploy tens of cards at single research laboratory site and then the cost of single card becomes significant.

IV. Evaluation of Synchronisation Methods

No actual signal exists that can be measured from a computer to determine if it is running with correct time. Consequently we used an indirect method to determine synchronisation accuracy. We set two computers to capture network traffic at the same point of network and then analysed the difference in time stamps of captured packets. We can use that to determine how well those two computers are in sync.

A traffic generator (Spirent Adtech AX/4000) was used to generate packets with fixed intervals on 100 Mbit/s Ethernet. The link goes through two 10/100 copper Ethernet taps and back to a traffic analyser. The traffic analyser part of AX/4000 was used to verify that every packet sent was also received. The two computers were connected to analyser outputs of Ethernet taps to ensure that both computers receive packets simultaneously.

On both computers, a `tcpdump` was run to record all packets. The 64-byte long packets were sent with fixed intervals, 10,000.04 packets per second. This rate was selected so that no packets were lost. To identify each packet when it was received, the source addresses of IP datagrams were incremented for each packet in periods of 1,024 packets. At post-processing the difference in time stamps between computers was calculated for each packet.

The computers used in the test were identical with Supermicro P8SCI motherboard, Intel Celeron D 2533/533 MHz processor with 256 KiB cache memory, 1 GiB memory and 200 GB SATA disk. The operating system was Linux 2.6.8.1

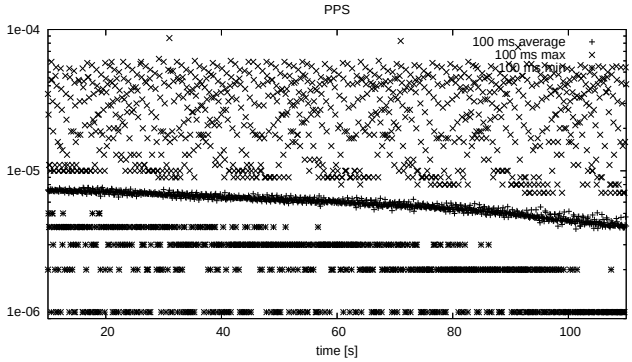

Fig. 6. Host synchronised by PPS to serial port and NTP over network. Time differences between packet received by two hosts

for NTP and PPS tests and Linux 2.6.13.3 for SynPCI test. We used a different kernel version because PPS patch was at that time only available to 2.6.5 kernel and would need more modifications for 2.6.13 than for 2.6.8.

A. PPS with NTP over Network

The NTP process (version 4.2.0) was configured to use two stratum 2 NTP servers for seconds numbering and PPS signal on serial port for seconds epoch. The computers were left to synchronise with each other for two days before test runs began. Although the computers achieved synchronisation of few microseconds, as reported by `ntpdate`, once some load was applied to machines the synchronisation was lost.

Figure 6 shows the maximum, average and minimum time difference calculated over 100 ms period. The mean time difference is 5.9 μs. We notice the average line to have a slight slope indicating that clocks run at different rate.

B. NTP over Network

For the second test, the NTP process was configured not to use PPS signal at all and to use only those two stratum 2 NTP servers. The computers were allowed to synchronise for half an hour and then a test run was done. The computers were initially synchronised with PPS signal, just the NTP process was restarted after configuration change. In that way, the conditions are a optimal for a network-only NTP.

It can be seen from Fig. 7 that NTP over network provides significantly worse synchronisation than when using PPS signal. The average time difference is more than ten times worse, 79 μs. Furthermore, a visible skew on mean value can be seen. If the measurement would have run for a longer period of time, the oscillations would have been more visible.

C. SynPCI

The systems were rebooted and initially wall clock time was synchronised with `ntpdate` to local (stratum 3) NTP server. After this SynPCI syncronised in a second to incoming signal, provideing accurate time values without need to wait for synchronisation. In addition to SynPCI modification, also

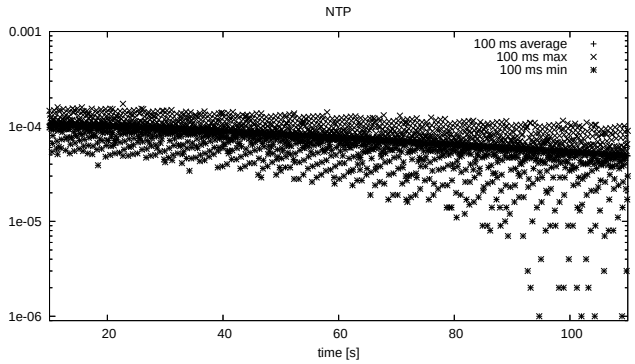

Fig. 7. Host synchronised by NTP over network. Time differences between packet received by two hosts

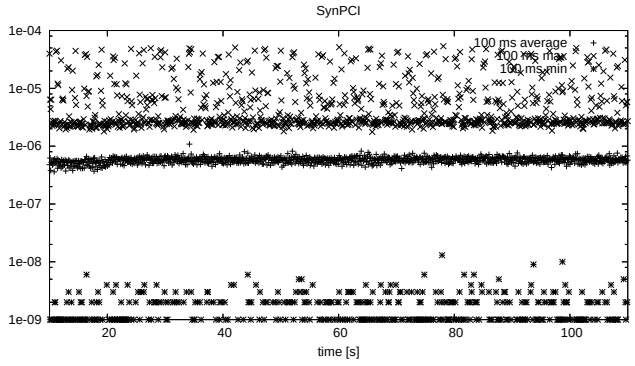

Fig. 8. Host synchronised by SynPCI over network. Time differences between packet received by two hosts

time stamp routine in kernel was modified to return values with nanosecond resolution.

As can be seen from Figure 8, the averages calculated over 100 ms periods is fairly constant while both PPS and NTP had some visible slope (refer to Figs. 6 and 7). Only two times over a 2 minute measurement period did the average exceed one microsecond. The average time difference is 0.61 μs.

A histogram showing time differences is presented in Fig. 9. We can see that SynPCI provides better performance than PPS. Using only NTP to synchronise results significantly worse performance.

V. Conclusion

Measuring synchronisation accuracy possesses a set of problems. If we consider PCI bus that has clock frequency of 33 MHz, each clock cycle shall take 30 ns and a single 32-bit read will use at least 5 clock cycles, or 150 ns. Thus it is very hard to improve clock resolution below microsecond.

Good synchronisation accuracy is hard to maintain under heavy system load as interrupt latencies increase. The approach used with our SynPCI-card helps to achieve good synchronisation accuracy under heavy load as it is insensitive to interrupt latencies. It is more expensive than using PPS on

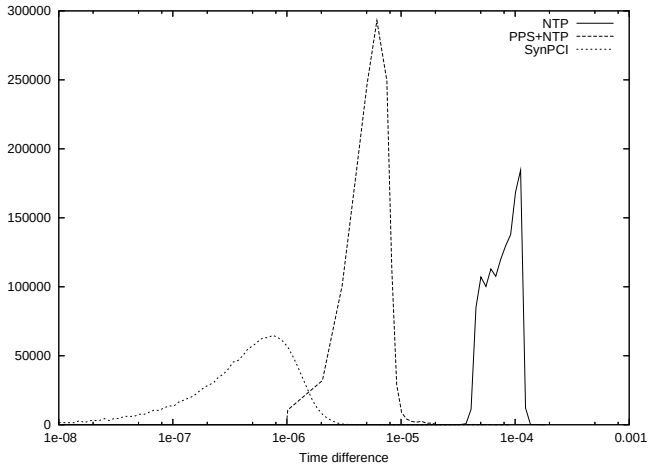

Fig. 9. A histogram of time differences between two hosts

serial port, but as our measurements indicated, PPS synchronisation does not work well under heavy load.

Using PPS to provide accurate syncronisation could be feasible if there are long periods of time with low prosessing load. One significant advantage of SynPCI is its ability to syncronise quickly. This is significant in prototyping environment where computers are frequently rebooted. One does not need to wait for clocks to syncronise.

Although one can buy, with significantly higher price, a special purpose capture card, those do not help on synchronising packet transmission or other time-critical applications like automation and control. Those capture cards have their value in providing accurate time stamps and zero-loss packet capture: something that is hard to gain with generic NICs. Interrupt moderation is good from throughput point of view whereas it makes timing accuracy worse. Nonetheless, keeping clocks synchronised removes a part of uncertainty from measured results as there won't be any clock skew. If delays measured from the network are larger than errors caused by interrupt moderation and congestion, then using SynPCI would provide stable measurements even over long periods of time.

Future work will include additional measurements to estimate true accuracy of SynPCI, testing multi-site installations and implementing drivers for other operating systems. A driver for FreeBSD is almost completed. The possibility to have accurate synchronisation in routing prototype network will help in studying different traffic control and routing methods. Using mixture of commercial and homegrown measurement tools gives us best of both worlds when we can run systems syncronised in time. With an accurate clock and synchronised systems one can focus one's studies into phenomenon, not irregularities of present implementation.

References

[1] S. Bregni, *Synchronization of Digital Telecommunications Networks*. John Wiley & Sons, Ltd., 2002.

[2] D. Mills, *Network Time Protocol (Version 3) Specification, Implementation and Analysis*, Mar. 1992. RFC 1305.

[3] D. Mills, *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI*, Oct. 1996. RFC 2030.
[4] J. Micheel, S. Donnelly, and I. Graham, "Precision timestamping of network packets," in *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, (New York, NY, USA), pp. 273–277, ACM Press, 2001.
[5] M. Kihara, S. Ono, and P. Eskelinen, *Digital Clocks for Synchronization and Communications*. Artech House, Inc., 2003.
[6] D. W. Allan, M. A. Weiss, and J. L. Jespersen, "A frequency-domain view of time-domain characterization of clocks and time and frequency distribution systems," in *Proceedings of the 45th Annual Symposium of Frequency Control*, 1991.
[7] F. Walls and J.-J. Gagnepain, "Environmental sensitivities of quartz oscillators," *IEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 39, Mar. 1992.
[8] J. Postel, *Daytime Protocol*, May 1983. RFC 867.
[9] J. Postel and K. Harrenstien, *Time Protocol*, May 1983. RFC 868.
[10] J. Mogul, D. Mills, J. Brittenson, J. Stone, and U. Windl, *Pulse-Per-Second API for UNIX-like Operating Systems, Version 1.0*, Mar. 2000. RFC 2783.
[11] IEC 61588 First edition 2004-09; IEEE 1588, *Precision clock synchronization protocol for networked measurement and control systems*, 2004.
[12] Range Commander's Council, White Sands Missile Range, New Mexico, *IRIG Serial Time Code Formats*, 2004. IRIG Standard 200-04.
[13] M. Carson and D. Santay, "Micro-time-scale network measurements and harmonic effects," in *Lecture Notes in Computer Science*, vol. 3015, pp. 103–112, Jan. 2004.
[14] V. Smotlacha, *Time Issues in One-way Delay Measurement*. PhD thesis, Czech Technical University, 2005.