# SynUTC – High Precision Time Synchronization over Ethernet Networks

Roland Höller

Institute of Computer Technology
Vienna University of Technology
Viktor-Kaplan Str. 2, 2700 Wiener Neustadt, Austria
tel +43-2622-23420 fax +43-2622-83423
e-mail: roland.hoeller@tuwien.ac.at

Günther Gridling

Department of Automation
Vienna University of Technolgy
Treitlstraße 1, 1040 Vienna, Austria
tel +43-1-58801-18325
e-mail: gg@auto.tuwien.ac.at

Martin Horauer

Technikum Vienna
Höchstädtplatz 3, 1200 Vienna, Austria
tel +43 1 3334077 291
e-mail: Martin.Horauer@technikum-wien.at

Nikolaus Kerö

Oregano Systems
Design & Consulting GesmbH
Phorusgasse 8, 1040 Vienna, Austria
tel +43-676-843104-300
e-mail: keroe@oregano.at

Ulrich Schmid

Department of Automation
Vienna University of Technology
Treitlstraße 1, 1040 Vienna, Austria
tel +43-1-58801-18325 fax +43-1-58801-18391
e-mail: s@auto.tuwien.ac.at

Klaus Schossmaier

Department of Automation
Vienna University of Technology
Treitlstraße 1, 1040 Vienna, Austria
e-mail: Klaus.Schossmaier@cern.ch

## *Abstract*

This article describes our SynUTC[*] (Synchronized Universal Time Coordinated) technology, which enables high-accuracy distribution of GPS time and time synchronization of network nodes connected via standard Ethernet LANs. By means of exchanging data packets in conjunction with moderate hardware support at nodes and switches, an overall worst-case accuracy in the range of some 100 ns can be achieved, with negligible communication overhead. Our technology thus improves the 1 ms-range accuracy achievable by conventional, software-based approaches like NTP by 4 orders of magnitude. Applications can use the high-accuracy global time provided by SynUTC for event timestamping and event generation both at hardware and software level.

SynUTC is based upon inserting highly accurate time information into dedicated data packets at the media-independent interface (MII) between the physical layer transceiver and the network controller upon packet transmission and reception, respectively. As a consequence, every node has access to the local time information of any communication peer and can therefore re-adjust its local clock accordingly. This enables both simple solutions based upon synchronizing with a (GPS-equipped) master node as well as elaborate fault-tolerant clock synchronization algorithms.

Each node must be equipped with a special network interface card (NIC) for this purpose, which extends standard NIC chipsets by a custom hardware encapsulated in a single IC plugged into the MII. This chip contains primarily a high-precision adjustable adder-based clock and timestamping registers as well as an uC core executing the synchronization algorithm. Our technology is generic, in the sense that our hardware support can be used with any NIC chipset based upon the MII. Moreover, since clock synchronization, except a simple service that broadcasts messages on a regular basis, is performed by the on-chip uC, standard NIC device drivers and protocol stacks can be used without changing. Last but not least, the principle underlying SynUTC is not limited to Ethernet-based networks but is applicable for any packet-oriented data network as well.

To verify the feasibility of our approach, a research prototype has already been developed and evaluated successfully. Currently, a multi-node demonstration system is being built to facilitate the transfer of our SynUTC technology in commercial applications.

[*] The SynUTC-project (http://www.auto.tuwien.ac.at/ Projects/SynUTC/) received support from the Austrian Science Foundation (FWF) grant P10244-ÖMA, the OeNB "Jubiläumsfonds-Projekt" 6454, the BMfWV research contract Zl.601.577/2-IV/B/9/96, and the Austrian START programme Y41-MAT. The present work was also supported by the Austrian "Gesellschaft für Mikroelektronik" (GMe).

## I. INTRODUCTION

Networked distributed systems benefit from a reliable, high-accuracy distributed time service in various ways. Apart from enabling high-precision simultaneous triggering of events and synchronous data acquisition at different nodes, tight clock synchronization is also advantageous for emerging networks destined to handle voice, data and Internet Protocol (IP) traffic: Maintaining an acceptable end-to-end quality of service in such networks requires continuous monitoring and maintenance of quite small latencies, which are caused by protocol processing, system transfer rates, frame forwarding mechanisms, etc. Clock synchronization accuracies in the ms-range, as provided e.g. by the well-known NTP time service, are not sufficient for this purpose.

Synchronizing an ensemble of distributed clocks comes in two flavours:

- Internal clock synchronization aims to keep the deviation between all clocks bounded, i.e., if $C_i(t)$ and $C_j(t)$ denote two fault-free clocks within a system, the worst case precision p satisfies $|C_i(t) – C_j(t)| = p\ \forall t = t_0$.
- External clock synchronization relates to the problem that clocks are required to follow an external reference like GPS time. The maximum deviation towards this reference time is called accuracy a, formally $|C_i(t) – t| = a\ \forall t = t_0$.

Reaching both goals jointly turns out to be a non-trivial problem, since a certain trade-off seems to be involved [5].

Pure software-based clock synchronization approaches do not take the medium access uncertainty at the sending node, any variable network delay, and the reception interrupt latency into account [8], [9]. The first one can be quite large for any network utilizing a shared medium, and the last one is seriously impaired by code segments with interrupts disabled. Typical values for the clock reading error reside in the range of several 100 µs to 1 ms.

With moderate hardware support developed along with our SynUTC project, we obtained results in the 1 µs range for both precision and accuracy [3]. With the integration of highly accurate timestamping features in the MII interface of Ethernet network interface cards (NIC) precision and accuracy can be further improved [4],[1],[2].

This paper surveys the current status of our technology. In Section II, the key features of our approach are described. Section III gives an overview of an entire system including an Ethernet switch. Section IV and V is devoted to the hardware support required at nodes and switches, respectively. A brief introduction to clock synchronization software issues is contained in Section VI. Some conclusions and directions of future work round off the paper.

## II. KEY OPERATING PRINCIPLES

Our approach does not use dedicated GPS-receivers per node, but disseminates time information via standard data packets. It thus avoids both fault-tolerance limitations and practical problems inherent in any "dedicated receiver"-solution:

- GPS-receivers do deliver wrong 1pps pulses now and then, and the large time-to-fix may cause a joining delay of 30 seconds or more for newly powered up nodes [7].
- Moreover, the "forest" of antennas required for a, say, distributed factory automation system with 100 nodes is difficult to accommodate and connect.

Our approach simultaneously increases the fault-tolerance degree and decreases the number of GPS receivers required in the system - without additional (cabling) costs, by using the existing data network only. The only price to be paid is some moderate hardware support and decreased precision/accuracy, which is hopefully acceptable for most applications.

Our SynUTC technology for high-accuracy clock synchronization is based upon a few key paradigms, which will be briefly introduced subsequently.

### A. *Adder-Based Clock*

Local time at any computing node is maintained by means of an unconventional adder-based clock [10], which uses a high-resolution adder instead of a simple counter for summing up the elapsed time between succeeding oscillator ticks. Owing to this, the clock can be paced by an oscillator with arbitrary frequency. Moreover, the local clock is fine-grained rate adjustable in steps of nsec/sec and supports state adjustment via continuous amortization as well as leap second corrections in hardware.

### B. *On-the-fly packet timestamping*

Clock synchronization over packet-oriented networks requires timestamping of data packets (Clock Synchronization Packets, CSPs) used for time transfer (see Figure 1). In purely software-based clock synchronization, timestamping at the sending resp. receiving side is done by reading the clock when assembling the CSP for transmission resp. in the packet reception interrupt service routine. This implies that the transmission delay uncertainty e includes both the network channel access uncertainty and the reception interrupt latency, which can be quite large. As a consequence, the achievable synchronization precision is quite poor (ms-range).

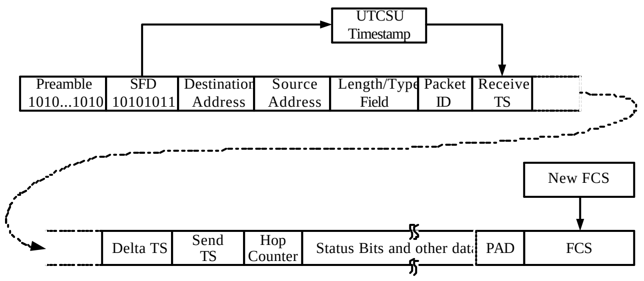

Figure 1: The structure of a clock synchronization packet (CSP). The packet ID allows management of additional features. The receive, delta, and send timestamps account for 96 Bits each. The 8 Bit hop-counter counts the number of switches along a CSP's path.

Our approach [3],[4] performs timestamping on-the-fly, when a CSP is sent resp. received by the network controller. As the Ethernet frame is received nibble by nibble via the MII interface, our special hardware scans the incoming data for the start frame delimiter (SFD) of the packet. With the clock cycle it encounters the SFD, the adder-based clock is triggered to store its current value in a 96 Bit timestamp register. This data is inserted in the "Receive TS" field of the CSP during reception or in the "Send TS" field of a CSP during transmission.

### C. Interval-based Paradigm

A unique feature of our approach is the support of the interval-based paradigm [6]. Real-time *t*, that is, GPS time or UTC, is not just represented by a single time-dependent clock value *C(t)* here, but rather by an accuracy interval ***C(t)*** that must satisfy *t* in ***C(t)***. Interval-based clock synchronization thus assumes that each node *p* is equipped with a local interval clock $\mathbf{C}_p$ that continuously displays *p*'s instantaneous accuracy interval $\mathbf{C}_p(t) = [C_p(t) - a_p^-(t), C_p(t) + a_p^+(t)]$ (see Figure 2). Naturally, $C_p(t)$ is just the local clock value and $\mathbf{a}_p(t)=[-a_p^-(t), a_p^+(t)]$ the negative and positive accuracy maintained by two additional adder-based clocks.

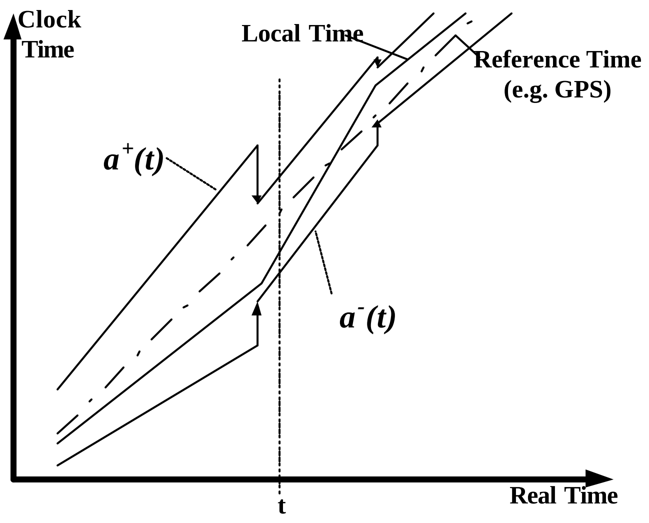

Figure 2: Basic issues of interval clocks and accuracy intervals. Local time follows the reference time within the accuracy bounds.

### D. Clock State and Rate Synchronization

An interval-based (external) clock synchronization algorithm is in charge of maintaining any node's local clock such that a worst case precision and accuracy is guaranteed [11]. Moreover, we synchronize not only the state of the clocks in the system, but also their progression: An additional interval based clock rate synchronization algorithm [12] is employed, which achieves high synchronization precision without expensive OCXOs as the nodes' frequency sources and frequent resynchronizations.

## III. SYSTEM HARDWARE OVERVIEW

The SynUTC prototype system consists of four network nodes, a COTS Ethernet switch, a time synchronisation add-on to the switch, and a GPS-receiver (see Figure 3). The network nodes are personal computers (PC/104-Plus) and are additionally equipped with the specialized network interface card, providing essential hardware support for the synchronisation mechanism. These network interface cards (NIC) use the PCI bus to communicate with the node's CPU.

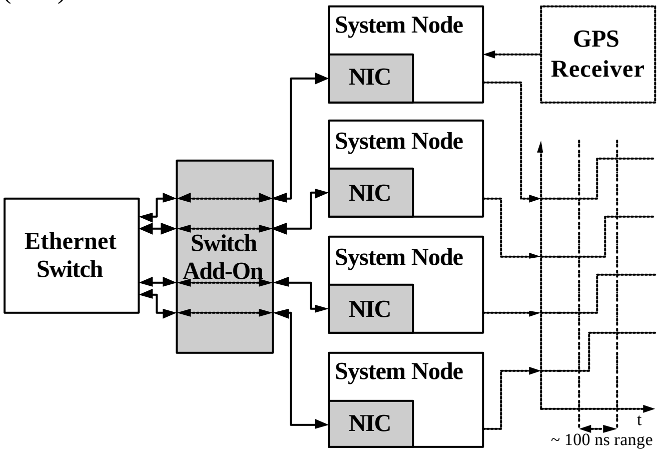

Figure 3: A SynUTC system for fault-tolerant external clock synchronization. The time synchronization system nodes are standard personal computers equipped with a special network interface card. All ports of the used COTS Ethernet switch are connected to the switch add-on, which timestamps bypassing clock synchronization packets.

The system nodes synchronize time via sending Clock Synchronization Packets (CSP) over the Ethernet LAN. CSPs use a special type field in the Ethernet packet header to be distinguished from all other network traffic. All data processing and calculations for the synchronization algorithm is performed by a 32 Bit microcontroller IP core, located on the NIC. The node's CPU does not see any difference between this NIC, which supports clock synchronization, and any other COTS Ethernet network card using the same Ethernet controller chip. Hence the whole node's system is unchanged except the fact that the node CPU has to broadcast CSPs on a regular basis (e.g. every ten seconds).

## IV. NETWORK INTERFACE HARDWARE ARCHITECTURE

Implementing a clock synchronization service according to the ideas outlined in the previous sections requires moderate hardware support on every node, which can be provided in a single custom chip (UTCSU-ASIC). It hosts the following functionality:

- A high-resolution adder-based local clock with a mechanism for linear continuous amortization.
- Two additional adder-based clocks holding and automatically deteriorating the bounds on accuracy with respect to external reference time.

- An interface to a GPS timing receiver, made up of a 1-pps digital input and a RS232 serial interface.
- Application-level event generation and time-stamping capabilities.
- A standardized interface for packet timestamping near the physical layer (e.g. IEEE 802.3 MII).
- A bus interface.
- An on-chip 32-bit microcontroller running the synchronization algorithm.

Figure 4 shows how the UTCSU-ASIC is integrated with the network controller on our NIC. Whenever a CSP is sent or received by the NIC, the UTCSU inserts a timestamp supplied by its adder-based clock.

This is actually achieved by manipulating the data stream at the Media Independent Interface (MII) between the physical layer device (PHY) and the Ethernet controller. Obviously, the packet's CRC and other error correction mechanisms are updated while the packet traverses the chip as well. The timestamps contained in a CSP are eventually used by the clock synchronization algorithm running on the on-chip microcontroller, which computes and applies suitable clock adjustment values.

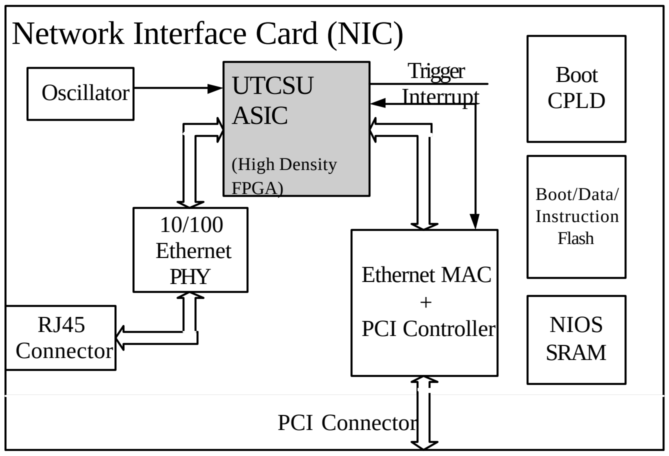

Figure 4: The network interface card with the clock UTCSU ASIC (NICA) placed into the media independent interface. The ASIC is equipped with a 32 Bit microcontroller, which executes the clock synchronisation algorithm. Boot CPLD, Boot/Data/Instruction Flash, and SRAM are needed for configuring the high density FPGA and for the integrated µC.

## V. SWITCH ADD-ON HARDWARE ARCHITECTURE

Recently, micro-segmentation by using switched Ethernet technology has become popular to handle all traffic in enterprise and industry networks. Supporting high-accuracy clock synchronization for these types of networks is hence mandatory.

As packets are sent from a node's NIC they will arrive at the prototype system's Ethernet switch for being forwarded to their destination address. Ethernet switches exhibit a substantial delay of typically several 10 ms for a packet having to pass through the switch, and – even worse – this delay varies over time. To overcome these unfavourable properties with respect to high-accuracy clock synchronization, a Switch Add-On is placed into the transmission path of the CSPs (see Figure 5). The Switch Add-On is equipped with a Switch Add-On ASIC (SAOA), which inserts time stamps into the CSPs upon reception and calculates the accurate amount of time the packet has spent on the Ethernet switch before it is sent to its destination. All packets from or to a node have to pass through the Ethernet switch and therefore through the Switch Add-On as well.

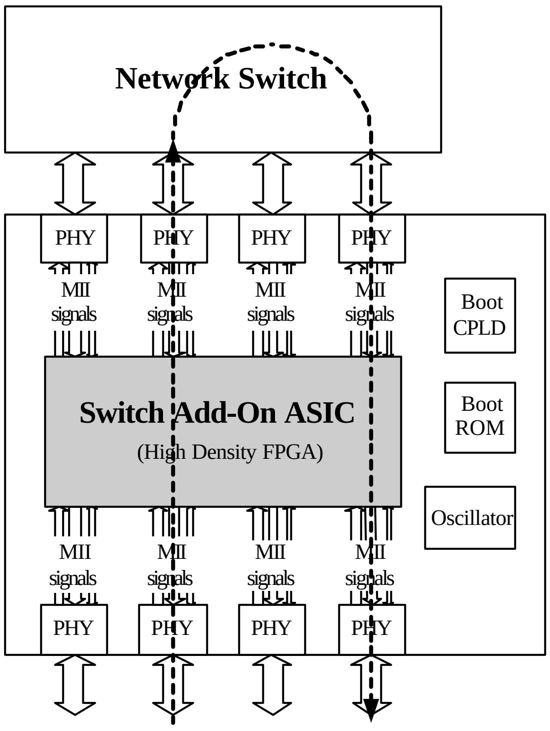

Figure 5: The switch add-on as it is connected to the COTS network switch. As CSPs traverse the switch add-on ASIC, which is realized as a high density FPGA, they are timestamped by the special clock synchronisation and timestamping hardware. Boot CPLD and Boot ROM are for FPGA configuration during power up. The oscillator is an OCXO to allow accurate measurement of the packets delay through the Ethernet switch.

## VI. CLOCK SYNCHRONIZATION SOFTWARE

The clock synchronization software is executed directly on the NICA by a 32 bit Altera NIOS IP core. Besides executing the synchronization algorithm itself, the software must also measure system parameters like the transmission delays and the clock drifts.

The synchronization algorithm is round-based; one round comprises the following activities:

- The full message exchange (FME), where every node broadcasts a clock synchronization packet (CSP) with its own clock value and accuracy interval to its peers.
- A message collection phase where the node waits for the messages of its peers.

- The computation of the new clock value and accuracy interval by a fault tolerant interval intersection function (FTI).
- The re-synchronization itself where the computed clock value is taken over.
- The free-run phase in which the clock simply runs without any interference from the algorithm.

The transmission of a CSP is periodically initiated by the host CPU. Timestamping is done on the fly by the NICA hardware, so the software only needs to collect the data from the packets passing through the NIC.

The time intervals contained in an incoming CSP need to be adjusted before they can be used in the FTI algorithm: The computation phase where the intervals are actually used is some time after the FME, so although the intervals a node has received were correct when they were sent, they already have deteriorated by the time they are used due to two effects:

First, we must consider the message delay which stems from transmitting the message on the network. This delay consists of a deterministic part d and an indeterministic part $e^{\pm}$. The actual delay $d_i$ of a message on the network (without switches) can be bounded by the interval [d] = [d - $e^-$, d + $e^+$] where the expectation $E(d_i) = d$. Delays added by switches are protocolled by the switch add-on and are handled separately to keep the indeterministic part of the message transmission delay as low as possible.

Secondly, since all oscillators deviate slightly from their ideal frequency as depicted in Figure 2, the software has to compensate for the clock drifts as well.

In consequence, the algorithm first adjusts the received intervals to reflect the current state of the remote clocks before using them for the intersection function. The accurate knowledge about transmission delays and clock drifts required for the adjustment is gained from periodic transmission delay and rate measurements. Like CSPs, these measurement packets are sent by the host CPU.

Neither clock synchronization nor system parameter measurement are very demanding in terms of hardware resources. In fact, a large part of the computational effort goes into handling the 96 bit integers representing the timestamps. However, if we add more sophisticated features like fault detection or support for partially connected networks, then the requirements in terms of memory and computation power increase considerably. Especially fault detection mechanisms tend to be quite demanding, which is why we do not implement any of these features in our first prototype.

## VII. Conclusion and Future Work

We presented an overview of our SynUTC technology, which facilitates high-accuracy time synchronization over Ethernet networks. Based upon a novel Media Independent Interface-based timestamping method in conjunction with clock rate synchronization, our approach improves the precision and accuracy achieved by software-based solutions like NTP by 4 orders of magnitude. All the hardware support required is provided by a single custom UTCSU ASIC, which contains digital circuitry for a high-resolution adjustable clock, timestamping mechanisms, the interface to a GPS receiver, and a microcontroller that executes the clock synchronization algorithm.

A second generation prototype implementation, which is currently being built, will be used for a long-term system evaluation. It will also be required for pushing our technology into industrial pilot applications, which became feasible by our approach.

## VIII. References

[1] Martin Horauer, Hardware Support for Clock Synchronization in Distributed Systems, Supplement of the 2001 International Conference on Dependable Systems and Networks, Göteborg, Sweden, 1-4 July 2001, pp. A-10 – A-13.

[2] M. Horauer, R. Höller, Integration of highly accurate Clock Synchronization into Ethernet-based Distributed Systems, SSGRR 2002w, 2002, ISBN 88-85280-62-5.

[3] Ulrich Schmid, et al., A Network Time Interface M-Module for Distributing GPS-time over LANs, J. of Real-Time Systems 18(1), 2000, pp. 24-57.

[4] Ulrich Schmid, Martin Horauer, Nikolaus Kerö, How to Distribute GPS-Time over COTS-based LANs, Proceedings of the 31st IEEE Precise Time and Time Interval Systems and Application Meeting, Dana Point, California, USA, December 1999, pp. 545-560.

[5] C. Fetzer, F. Cristian, Integrating External and Internal Clock Synchronization, Journal of Real-Time Systems, May 1997, No. 3, Vol. 12 (2), pp. 123–172.

[6] U. Schmid and K. Schossmaier, Interval-based clock synchronization, Journal of Real-Time Systems, May 1997, No. 3, Vol. 12 (2), pp. 173– 228.

[7] D. Höchtl, U. Schmid, Long-Term Evaluation of GPS Timing Receiver Failures, Proceedings of the 29th IEEE Precise Time and Time Interval Systems and Application Meeting (PTTI'97), Dec. 1997, pp. 165-180.

[8] D. L. Mills, Internet time synchronization: The network time protocol, IEEE Transactions on Communications, October 1991, pp. 1482-1493.

[9] D. L. Mills, Improved algorithms for synchronizing computer network clocks, IEEE Transactions on Networking, June 1995, pp. 245-254.

[10] K. Schossmaier, U. Schmid, M. Horauer, D. Loy, Specification and Implementation of the Universal Time Coordinated Synchronization Unit (UTCSU), Journal of Real-Time Systems, May 1997, pp. 295-327.

[11] U. Schmid, Orthogonal Accuracy Clock Synchronization, Chicago Journal of Theoretical Computer Science, 2000, pp. 3-77.

[12] K. Schossmaier, An Interval –Based Framework for Clock Rate Synchronization Algorithms, Proceedings 16th Symposium on Principles of Distributed Computing, August 1997, pp. 169-178.