Highly Accurate Time Synchronization over Switched Ethernet

Tor Skeie[a]
ABB Corporate Research
Bergerveien 12
N-1375 Billingstad
Norway

Svein Johannessen
ABB Corporate Research
Bergerveien 12
N-1375 Billingstad
Norway

Øyvind Holmeide[b]
ABB Corporate Research
Bergerveien 12
N-1375 Billingstad
Norway

a. Also affiliated with the Department of Informatics, University of Oslo. E-mail: tskeie@ifi.uio.no
b. Also affiliated with OnTime Networks. E-mail: oeyvind@ontimenet.com

***Abstract* – In the automation world the intelligent electrical devices need to be accurately synchronized for time stamping of data and motion control. In this paper we discuss how to achieve precise time synchronization over switched Ethernet networks in a generic way. The addressed synchronization requirements relate to substation automation, which is associated with five levels of synchronization accuracy defined as the IEC 61850 classes T1 to T5. The paper challenges the hardest requirements and present general solutions for IEC classes T4/T5 (4µs/1µs) and IEC class 3 (25 µs) in multi-traffic LAN environments. The classes T4/T5 solution is based on an Ethernet switch from OnTime Networks, while the class T3 solution relies on standard Ethernet switches. Common for both solutions is that they adhere to a low-level time stamp implementation of the Simple Network Time Protocol.**

I. TIME STAMPING AND SYNCHRONIZATION

Time stamping is the association of a data set with a time value (in this context, "time" may also include "date").

When several nodes are connected together in any sort of network, a requirement is that they should be *synchronized* (show the same time at one point in time). One solution is to elect one node to be the "time reference", which means that every other node should get the current time from it at least once a day and set its own clock to agree with that time. This solution works satisfactorily on a local area network (LAN), but all node clocks will lag the time reference by the time it takes a clock value to travel from the time reference to the synchronizing node. Except for very unusual cases, this lag is less than one second and thus good enough for office purposes.

Enter the Internet. Suddenly the synchronization problem escalates, since two collaborating nodes may be located in different time zones (remember to compensate for that) and a synchronization message may take a long time from one node to the other. Fortunately, the Internet Network Time Protocol has a solution to both problems. This protocol involves sending a time stamped time request message to a "timeserver". This timeserver adds an arrival time stamp and a retransmit time stamp before returning the request message to the requesting node. The requesting node time stamps the message when it returns and uses all the time stamps in calculating the correct time. This protocol, and the related Simple Network Time Protocol, is able to synchronize computers across the Internet with a precision in the low milliseconds.

II. AUTOMATION AND THE NEED FOR NETWORKING

While office computers were getting networked left and right, traditional automation and measurement systems were far slower to accept this new paradigm. One of the reasons for this delay was that such systems usually depends on being able to sample input data at equally spaced points in time. The "field bus" concept was the first step on the road to networked automation systems. Usually marketed as a cost-saving device (cabling being the main outlay in any automation setup), the different types made different compromises between tight centralized control and network-type flexibility. If you wanted precise control over the data-sampling task, you either implemented a field bus with tight centralized control or you "cheated" by using a separate wire for controlling the input data sampling.

The problems with such field bus implementations were twofold. Firstly, the field busses had a much smaller market than the office network, and therefore the price remained fairly high. Secondly, their data throughput capacity was much lower than that of the office network where Fast Ethernet (or even Switched Fast Ethernet) had become a de facto standard at a very competitive price. Therefore, using Fast Ethernet for automation purposes looked like a sure winner – if you could just solve some practical problems:

- Ethernet is by nature "democratic" – all nodes have an equal chance of accessing the network at a given point in time. Automation systems are "dictatorial" – some nodes are more important than others and want to have priority when accessing the network.
- Ethernet is by nature "statistic" – there is no way of predicting just when a queued message will actually arrive at the destination. Automation systems want assurance, not probabilities.
- Ethernet is a full-fledged communication network with a fairly large overhead – small data packets (like the ones in traditional data collection) are grossly inefficient.

Most of these problems are being solved these days. IEEE 802.1p and IEEE 802.1Q address the need for priority across the network. Switched Ethernet goes a long way towards guaranteeing a maximum message delivery time. Intelligent nodes with local real-time clocks sample their data at predefined points in time, time stamp the data, pack several data sets into a network packet and transmit the data at their leisure. The only thing we lack is the connection between the local clock and that in the central controller.

III. STATING THE PROBLEM – WHY NETWORK SYNCHRONIZATION IS DIFFICULT

The delays from the time stamping of a time synchronization message in the message source node until it is time stamped in the message destination node are:

- Message preparation delay
- Communication stack traversal delay (transmission)
- Network access delay
- Network traversal delay
- Communication stack traversal delay (reception)
- Message handling delay

Variations in the delays are due to:

- Real-time OS scheduling unpredictability
- Network access unpredictability
- Network transversal time variations

Time stamping at the lowest stack level helps eliminate the stack delay variations and real-time OS scheduling unpredictability but introduces some complications in the implementation. We will discuss the delays and delay variations later, but first we will take a look at the requirements.

IV. SYNCHRONIZATION REQUIREMENTS IN SUBSTATION AUTOMATION

In the energy distribution world, a *substation* is an installation where the energy is combined, split or transformed. A Substation Automation (SA) system refers to tasks that must be performed in order to control, monitor, and protect the primary equipment of such a substation and its associated feeders. In addition, the SA system has administrative duties such as configuration, communication management and software management.

The communication within SA systems is crucial from the point of view that the functionality demands a very time-critical data exchange. These requirements are substantially harder than the corresponding requirements in general automation. This is also true for the required synchronization accuracy of the IED's[1] internal clock in order to guarantee precise time stamping of current and voltage samples. Various SA protection functions require different levels of synchronization accuracy; in this respect IEC has provisionally defined five levels - IEC classes T1 – T5 (IEC 61850-5, sections 12.6.6.1 and 12.6.6.2 [4]):

- IEC class T1: 1 ms
- IEC class T2: 0.1 ms
- IEC class T3: ±25 µs
- IEC class T4: ±4 µs
- IEC class T5: ±1 µs

Since these definitions and classes are not frozen as yet, we will refer to them as class T1, class T2 etc. (without IEC) in this article

Observe that these numbers represent the required tolerances. Later in the article we will discuss the impact these numbers have on local clock resolution and accuracy.

Towards Open Solutions

At this point in time, the SA business is on the edge of migrating towards open solutions. A proof of this new trend is the upcoming IEC 61850 standard on *Communication networks and systems in substations* issued by Technical Committee 57 [4]. The vision is to achieve interoperability between products from different vendors on all levels within the substation automation field. Inventions of de-facto standard concepts and adoption of off-the-shelf technologies are the key instruments to reach the interoperability goal. In order to reach the vision of interoperability there must also be consensus within IEC concerning the communication medium and protocols to be deployed on the various levels in a substation.

The high-speed properties of Ethernet together with its dominant position in LAN networking makes Ethernet the prime communication candidate for substation automation use [7]. Several studies have shown that *Switched Fast Ethernet* has sufficient real-time characteristics to meet SA demands [5][13]. What is left to show is that it is possible to implement the various classes of synchronization accuracy over *Switched Fast Ethernet*. Such a step would eliminate the need for the dedicated (separate) links that are used for this purpose today. Those links are considered to the final obstacle of fully migrating to Ethernet in SA.

Figure 1 illustrates the communication structure of a future substation adhering to switched Ethernet as a common network concept holding multiple coexisting traffic types.

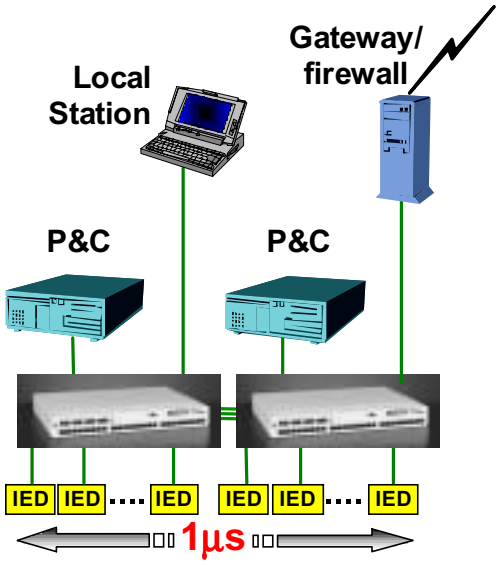

Figure 1 A future substation using Switched Fast Ethernet as the common single network infrastructure (P&C stands for Protection and Control).

1. IED: Intelligent Electrical Device

Related Work in the Network Time Synchronization Area

There is a plethora of proposed theory and methods for synchronizing clocks in distributed systems [1][2][3][6][8][10][11]. The most prominent public domain synchronization method is the Network Time Protocol (NTP) proposed by Mills and Internet Engineering Task Force (IETF) group [8]. A subset of NTP (SNTP - Simple Network Time Protocol) [12], is also defined, and is protocol-compatible with NTP. The intended use of NTP is to synchronize computer clocks in the global Internet. For this purpose it relies on sophisticated mechanisms to access national time, organize time synchronization subnets possibly implemented over various media, and adjustment of local clock in each participating peer. SNTP on the other hand does not implement the full set of NTP algorithms and is targeting simpler synchronization purposes.

Common for this body of work is that it does not present solutions for low microsecond requirements, instead targeting synchronization of LANs and WANs in the general sense where a precision of some milliseconds is sufficient. There is one exception, however, in [9] Mills describe engineered refinements in the form of modified driver and Unix kernel code for improving the time stamping accuracy within NTP time clients (servers). By these adaptations a timekeeping precision of few hundred microseconds for an Ethernet network of workstations is reportedly practical. From the proposed NTP implementations discussed here it follows that the accuracy requirements of the classes T1 and T2 are within reach.

Looking at the automation field in general and especially at the SA world, we find a diversity of proprietary and patented solutions in order to achieve highly accurate time synchronization over Ethernet. *ABB* holds several patents[2] in the time synchronization area. These are based on the concept of a Time Master broadcasting a *tick* (a special network message) and directly afterwards broadcasting a Time Synchronization message stating when the tick left the Time Master. This two-step approach effectively removes the real-time OS scheduling unpredictability and the network access unpredictability. U.S. Philips holds a patent[3] on the usage of hardware-based Ethernet time synchronization and clock adjustments being able to precise time tag Ethernet packets. Hewlett-Packard has a patented (possibly still pending) high-resolution time synchronization scheme (implemented in the BFOOT series of embedded controllers)[4]. The BFOOT chips are reportedly able to perform synchronization accuracy (one of the BFOOTs takes the role as master node) in the range of 200ns; however, this is over a single repeater hub network only.

V. BEING EXTREMELY ACCURATE – CLASS T3

We have mentioned that the precision that may be achieved by the traditional NTP/SNTP implementations is one millisecond at best. Basically, this stems from the time stamping of incoming and outgoing NTP/SNTP packets at the NTP/SNTP application layer (communication stack traversal delay with RTOS scheduling unpredictability). By special refinements of the driver and the OS kernel software, Mills reports an attainable accuracy of a few hundreds of microseconds. In this section we describe how 25 microseconds accuracy can be achieved based on a SNTP implementation and standard Ethernet switch interconnecting IED time clients and a timeserver. We focus on SNTP with some filtering techniques instead of the full NTP implementation, since the phase lock loop/filtering mechanisms of NTP may not be appropriate for a well-arranged substation automation network. The NTP filtering techniques are meant to cover variable delay through the protocol stacks and several network elements in e.g. a WAN, while we look into a LAN based infrastructure with only one Ethernet switch between the timeserver and the client. The timing accuracy degradation through the protocol stacks is more or less removed, and the design of the proposed filtering techniques is based on knowledge of the switch latency characteristics. The proposed filtering techniques are further described in VI.

A Tuned SNTP Time Protocol Implementation

The nature of real-time operating systems (they guarantee a maximum response time for an event but allow for a wide variation below that) introduces a substantial variation in the time spent in the communication stacks. This fact has necessitated an interrupt-level time stamping both in the time client and timeserver (this agrees with the findings of Mills in [9]). The class T3 solution described here adheres to the principle of *interrupt-level time stamping of the SNTP request packet when sent from the time client and when received at the timeserver*. Moreover, we propose that the synchronization should be based on the *transmit time stamp* set by the client (referred to as *t1* in SNTP terminology) and the *receive time stamp* set by the server (referred to as *t2*). Using a low-level transmission time stamp of the corresponding SNTP reply packet (referred to as *t3*) needs novel techniques for controlling the non-deterministic access of an Ethernet packet to an Ethernet bus. We leave this subject for the class T5 discussion where a patented method will be presented. A side effect of only using *t1* and *t2* in the calculations is that no mechanisms for automatic calibration of the network latency will be available and therefore a manual calibration of the propagation delays of the drop links and the minimum switch latency must be performed. illustrates the setup of a SNTP time client and timeserver implementing interrupt-level time stamping.

SNTP specifies time distributions based both on unicast and broadcast messages. In this paper we will only consider the unicast synchronization, which means that the time client controls the interval between two time requests (broadcast distribution of time is only based on transmission of SNTP reply packets from the server to the clients).

Now we shall take a look at the implementation of the time client, the timeserver and the Ethernet infrastructure, all

2. NO307728: Method for Providing Time Synchronization in a Network
   NO307768: Enhanced Accuracy Time Synchronization
   NO307769: Ethernet Time Synchronization
3. U.S.Patent 4,815,110: Method and a system for synchronizing clocks in a bus type local network
4. See http://www.hpie.com/news.htm

of which contribute to the overall error budget of the SNTP time synchronization loop.

Time Stamping implementation issues

In the time client three different low-level time stamping methods are evaluated. We will prove that only the two first ones are suitable for very accurate time synchronization:

1. Hardware time stamping in the Ethernet controller.
2. Software time stamping in an Interrupt Service Routine (ISR) outside the RTOS. This ISR should be connected to the Ethernet Interrupt Request signal and have a top hardware priority.
3. Software time stamping in an Interrupt Service Routine (ISR) controlled by the RTOS (Ethernet driver). This ISR is connected to the Ethernet Interrupt Request signal with a normal hardware priority.

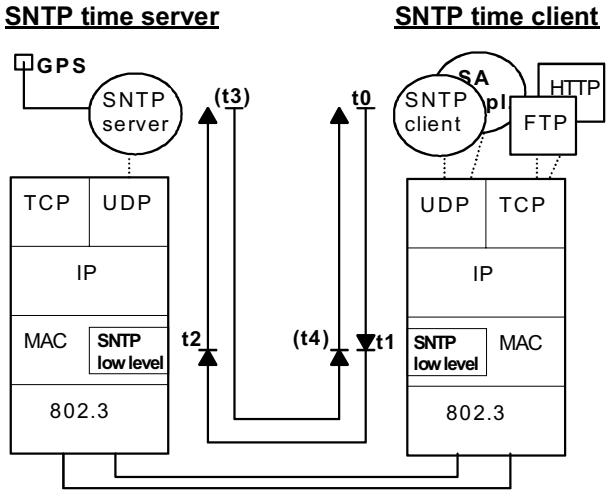

Figure 2 SNTP time client – server relation using low-level time stamping.

Using any of this low-level time stamping methods is considered an implementation issue and will not cause any incompatibly between a low-level time stamping client and a standard high-level time stamping server. In addition to low-level time stamping the time client must consider the following aspects:

- The interval between time updates.
- The specifications of the local time-of-day clock with respect to resolution, accuracy/stability, and the availability of drift and offset correction mechanisms.
- The usage of adaptive filtering and time stamp validation methods in order to remove network delay variations. These will be further discussed in the section *Class T3 time synchronization accuracy*.
- In the process of realizing the next generation time synchronization in substation automation, the time client will be implemented by the SA providers, while the timeserver presumably can be a third-party device.

Local Clock Implementation Issues

A recurring topic in real time clock discussions is the required *resolution* of the clock (often confused with the clock accuracy which is quite another matter). In our case it translates to: Which frequency should be the input to the real time clock when our precision requirements are 25 µs?

If we are measuring very precise events, a clock resolution of 25 µs would mean that we would time stamp an event somewhere between 0 and 25 µs after it happened. If our events are somewhat imprecise, such a resolution would mean a positive chance that we would be outside the limits. Since an Ethernet packet arriving on the drop link is first synchronised to the physical layer clock (25MHz or 2.5MHz) and then resynchronised to the processor clock, the precision of the packet arrival time is somewhere between 40 and 400ns. Given these two bounds and noting the fact that high frequency local clocks and time counters are easy to manufacture, we settled on a 10MHz clock source for the measurements. The corresponding measurement *precision* is calculated by taking the standard deviation of the measurements and observing that six standard deviations (three to each side) is a very good (99.5%) estimate of the time stamp precision.

Timeserver implementation issues

In order to achieve class T3, the timeserver should be able to time stamp an incoming message with an accuracy of better than 2µs independently of network load. The timeserver under test used hardware time stamping (see *Time Stamping implementation issues*). The exact time should be taken from a GPS receiver, and the time parameters distributed from the timeserver should be based on GPS time representation instead of absolute time (i.e. UTC timing) in order to cope with the leap seconds problem. It is also convenient if the timeserver supports full duplex connectivity in order to avoid a situation where upstream data introduces extra switch latency in downstream data (i.e. time requests).

Ethernet infrastructure implementation issues

Only one switch should preferably be allowed between a time client and a timeserver. Having multiple switch levels will impose increased jitter[5] through the infrastructure, which again might call for more complex filtering at the time client side. The Ethernet switch must also have good switch latency characteristics. The switch latency from the client drop link to the server drop link depends on several parameters:

- *General switch load*; this means all the network load on the switch except for the packets sent to the timeserver. The variations in the switch latency from the client drop link to the server drop link should be less than 2µs.
- *Timeserver load;* this parameter means other packet sent to the timeserver that may introduce extra delay in the transmission of a given SNTP request packet. This delay

5. Jitter=variations in the delay

can be handled at the time client side using various filtering techniques (see IV.).

- *Store-and-forward or cut-through*; most switches are based on "store-and-forward" technology[6]. This is a static parameter for a given switch. The basic switch delay is specified by the vendor and can therefore easily be incorporated in the calculations.

Measurements on an Actual Network

The test setup used in the verification of the SNTP server and client implementation is shown in Figure 3. The timeserver is based on the SNTP timeserver from OnTime Networks[17], while the SNTP client is implemented on an ARM7TDMI processor running the VxWorks real time operating system (RTOS).

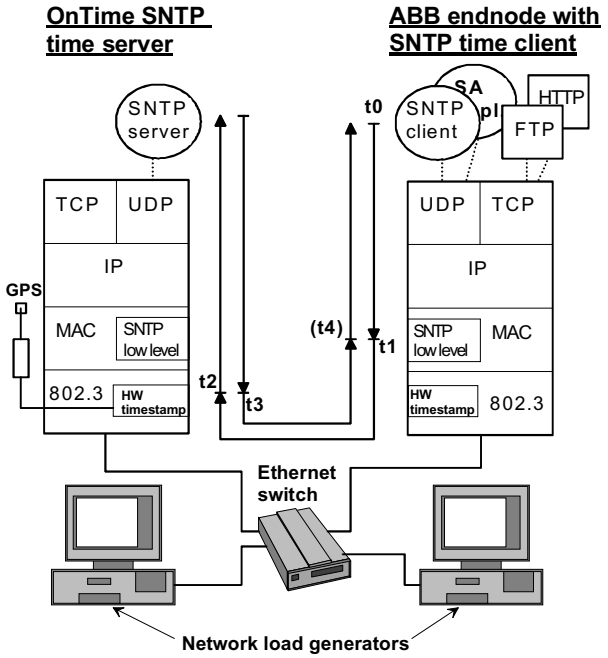

Figure 3 The SNTP experiment setup.

Results from Ethernet switch experiments

In this section we present the results of latency measurements on a switch from Cisco [16], Nortel [15] and 3Com [14] under various load conditions. These results will indicate the jitter characteristics that can be expected in the Ethernet infrastructure. Furthermore, the same characteristics shall also be observed in the full setup measurements when performing low level time stamping that substantially will eliminate the end node inaccuracy.

The following network load cases in addition to the time synchronization traffic were tested:

1. No load.
2. General switch load – i.e. 20000 Packet Per Second (PPS) of 500 bytes (90 Mbps) sent or received on each port except for the ports used for switch latency measurements.
3. Timeserver load – i.e. 50 % of the bandwidth of timeserver drop link is filled with dummy packets of 100 bytes each. The timeserver drop link is 10BASE-T; therefore the duration of the dummy packet is 80 µs.

The results are presented as a latency frequency distribution, where the latency through the switch is measured with a HP logic analyzer. The frequency distribution for the switch latency consists of the distribution for the no load scenario plus a distribution of queuing delays.

The distribution in Figure 6 is characterized by an average of 134.7µs, a standard deviation of 32.3µs and a median of 125.6µs. The standard deviation indicates a measurement accuracy of ±97µs.

From these experiments we can draw two conclusions:

- Traffic not destined for the timeserver does not interfere with traffic to the timeserver. This can easily be seen by comparing Figure 4 to Figure 5. The comparison indicates a proper design of the switching engine[7].

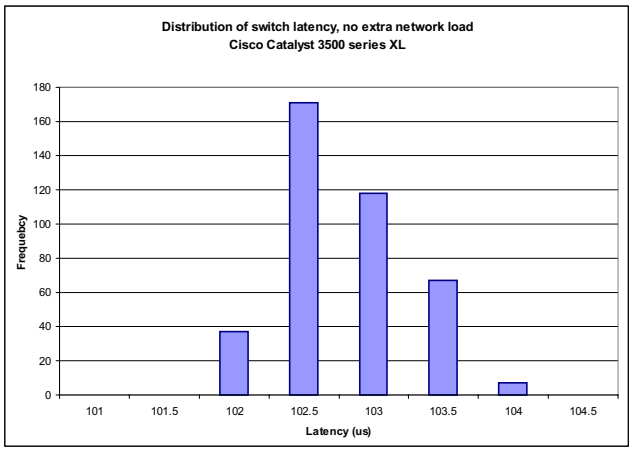

Figure 4 CISCO Catalyst 3524XL - no load.

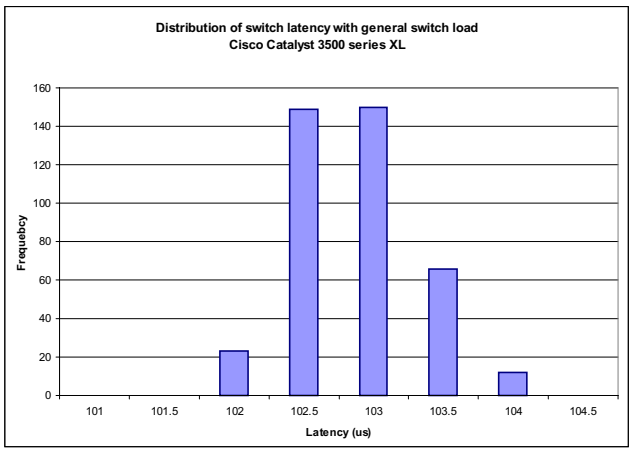

Figure 5 CISCO Catalyst 3524XL - general switch load

6. "Store-and-forward" means that the whole Ethernet packet is received (and CRC checked) before it is passed through the switch fabric. This delay depends on the length of the packet.

7. This conclusion might not true for all Ethernet switches on the market. The switches we have tested are all high-end switches based on the latest switch architecture. Thus, there might be higher variations in the switch latency on primitive low-end switches due to general switch load.

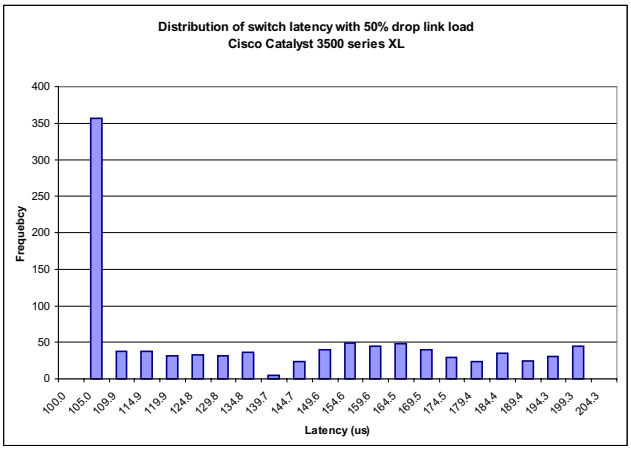

Figure 6 CISCO Catalyst 3524XL – 50% timeserver drop link load

- The switch latency for Ethernet packets to the timeserver depends to a great extent on other traffic to the timeserver.

Time Stamping Dependencies Within the Client

In this section we will discuss the accuracy of the various options for extreme low-level time stamping within the time client, where it was loaded with 350 interrupt requests per second. In general terms this represent lightly loaded conditions, but is should be mentioned that the ARM7TDMI based client is not a very powerful platform.

The precision attained when using a hardware time stamping scheme is, of course, independent of the software load in the Time Client node. Such a mechanism is, however, not present in commercial Ethernet controllers and we therefore want to know how precisely the other kinds of time stamping tracks the hardware time stamp. Figure 7 shows a distribution of the time differences between a hardware time stamp and the time stamp generated by an interrupt outside the RTOS under a high node processor load. This distribution can be summed up as follows:

- It is virtually independent of the node processor load.
- The average difference is 14.97µs, the median of the difference is 15.0µs and the standard deviation is 0.57µs. The standard deviation indicates a measurement accuracy of ±1.75µs.
- By hardware time stamping or low-level software time stamping outside the RTOS we have eliminated the time client inaccuracy in the error budget.

Figure 8 shows a distribution of the time differences between a hardware time stamp and the time stamp generated by a standard interrupt under RTOS control for the given processor load. This distribution can be summed up as follows:

- It is very dependent on the node processor load.
- The average difference is 69.2µs, the median of the difference is 61.0µs and the standard deviation is 21.0µs. The standard deviation indicates a measurement accuracy of ±63µs.
- Since the median differs from the average by about 10%, we cannot assume that the distribution is normal.

From these measurements we can conclude that:

1. Time stamping using a sufficiently high priority interrupt (preferably non-maskable) is for all practical purposes indistinguishable from time stamping using special-purpose hardware.
2. Time stamping using an interrupt under RTOS control needs sophisticated filtering and statistical techniques before it can be used for time synchronization purposes. In that respect this time stamping method is not suitable for class T3 synchronization

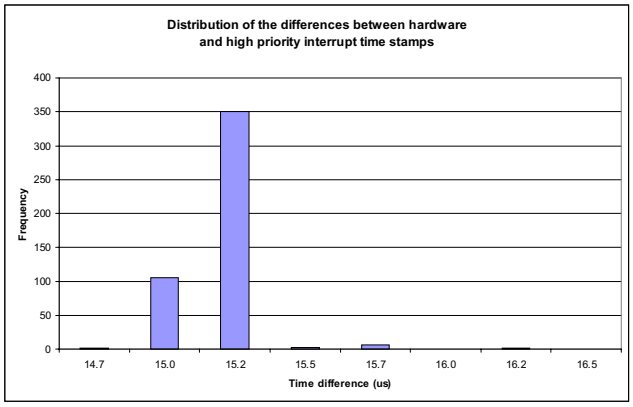

Figure 7 Distribution of the differences between time stamping in hardware and time stamping in an interrupt routine outside RTOS..

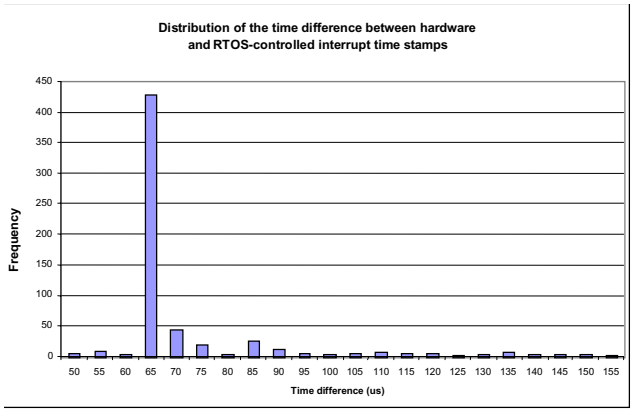

Figure 8 Distribution of the differences between time stamping in hardware and time stamping in a RTOS controlled interrupt routine.

VI. CLASS T3 TIME SYNCHRONIZATION ACCURACY

In the previous section we have shown that two different low-level time stamping methods may be used in order to eliminate the internal time client delays from the SNTP error budget. Since the timeserver relies on the same time stamping techniques, the only remaining factor to be handled in the error budget is possible time delay variations within the infrastructure. Below we discuss the procedures for adjusting the local clock in order to achieve the required precision under these circumstances.

Correcting for frequency deviation

If we look closely enough, the frequency of the local time-of-day clock will always differ slightly from the frequency of the reference clock in the timeserver. Finding that deviation and correcting for it is the first task in the synchronizing process. In a stable network it would only be to calculate the difference between the local clock (*t1*) and the timeserver clock (*t2*) twice, and see whether it increases or decreases and adjust the local clock frequency accordingly. However, due to the non-deterministic switch latency caused by head-of-line blocking, the *t2* time stamp may have an associated jitter, necessitating the use of statistical/filtering/erasure methods. Consider once more where the distribution of the heavy load switch measurement is shown. From this distribution we see that the switch latency consists of two parts:

1. Distribution equal to the no load scenario, refer Figure 4. This distribution has a small standard deviation of 0.4 µs. indicating a measurement accuracy of ±1.2µs.
2. More or less uniform distribution due to the extra load on the timeserver drop link.

One possible algorithm that may be used to estimate the client frequency deviation is as follows:

- Acquire a (statistically large) set of *t1*/*t2* pairs
- Using two *t1*/*t2* pairs at a time, calculate an estimate of the ratio between the reference clock and the client clock. Some of these estimates will be incorrect due to switch delay (caused by head-of-line blocking) but a large part of the estimates will be correct. The average/median of the distribution of the frequency ratios may then be applied as a measure of the frequency ratio between the reference clock and the client node clock.

The frequency deviation calculating procedure is below deployed for various loads on the server drop link. In these experiments 1000 *t1*/*t2* pairs were acquired in order to find the median of the frequency deviation estimation. Furthermore, we show the associated distribution of *t2*-*t1* adjusted for frequency deviation. This distribution will be the input for adjustment of absolute time (next section).

Figure 9 shows the distribution of the frequency deviation estimation while Figure 10 presents the corresponding distribution of *t2*-*t1* adjusted with the estimated frequency deviation in case of no load on the server drop link. From we see that the frequency deviation is easy to calculate and that the deviations from the estimated value are small under ideal conditions. The distribution given in is characterized by an average of 131.4µs, a standard deviation of 0.3µs and a median of 131.4µs. The standard deviation indicates a measurement accuracy of ±0.9µs.This distribution is similar to the one calculated from the corresponding switch measurements (Figure 4).

Figure 11 shows the distribution of the frequency deviation estimation and Figure 12 displays the corresponding distribution of *t2*-*t1* adjusted with the estimated frequency deviation in the case of 25% extra load on the server drop link. It follows from Figure 11 that it is easy to find the correct frequency deviation of the client with fairly good accuracy. The distribution in Figure 11 is characterized by an average of -0.23ppm, a standard deviation of 0.13ppm, and a median of –0.23ppm. The standard deviation indicates a measurement accuracy of ±0.4ppm.

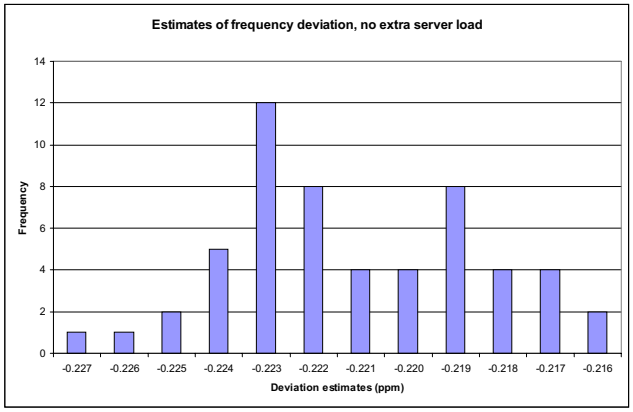

Figure 9 Estimated frequency deviation at no extra timeserver load

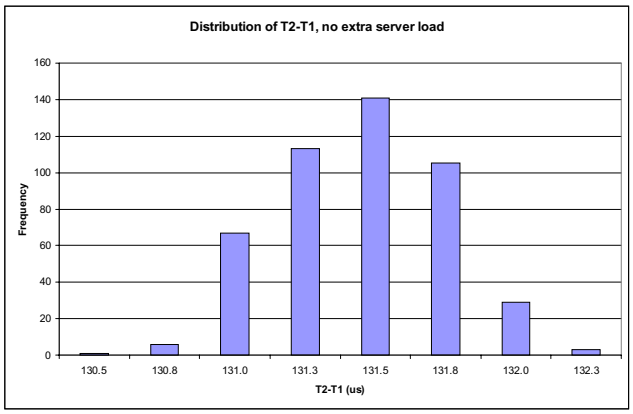

Figure 10 The differences between calculated and given values at no extra timeserver load

The distribution in Figure 12 is characterized by an average of 48.6µs, a standard deviation of 25.7µs and median of 36.1µs. The standard deviation indicates a measurement accuracy of ±77µs.

Figure 13 shows the distribution of the frequency deviation estimation and Figure 14 displays the corresponding distribution of *t2*-*t1* adjusted with the estimated frequency deviation in the case of 50% extra load on the server drop link. shows that it is still possible to find the correct frequency deviation of the client, but the precision suffers to some extent. The distribution in Figure 13 is characterized by an average of -0.22ppm, a standard deviation of 0.17ppm and median of -0.21ppm. The standard deviation indicates a measurement accuracy of ±0.5ppm.

The distribution in Figure 14 is characterized by an average of -31.8µs, a standard deviation of 32.0µs, and a median of -46.1µs. The standard deviation indicates a measurement accuracy of ±96µs. This distribution is similar to the distribution taken from the corresponding switch measurement (Figure 6).

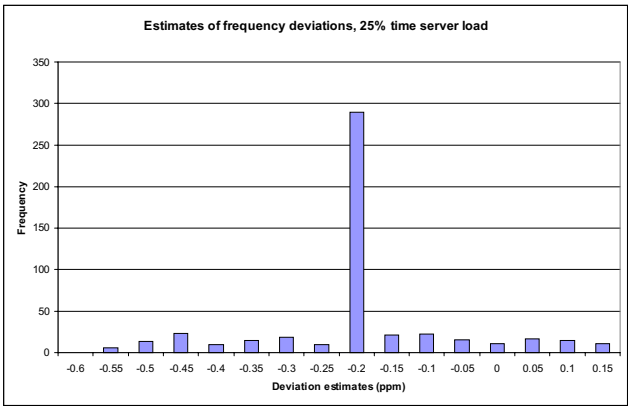

Figure 11 Estimated frequency deviation at 25% extra timeserver load

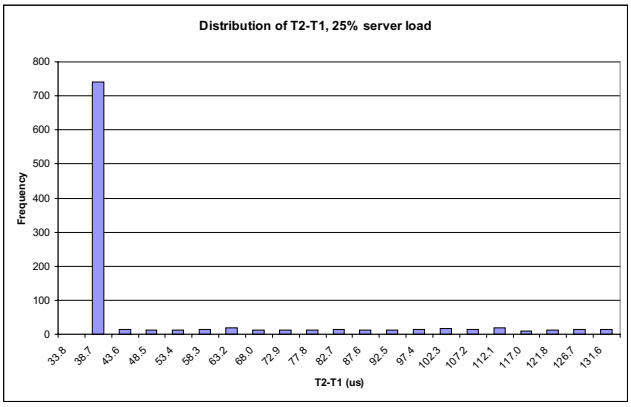

Figure 12 The differences between calculated and given values at 25% extra timeserver load

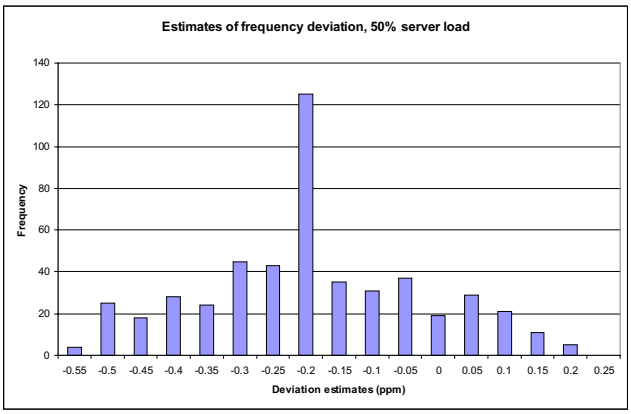

Figure 13 Estimated frequency deviation at 50% extra timeserver load

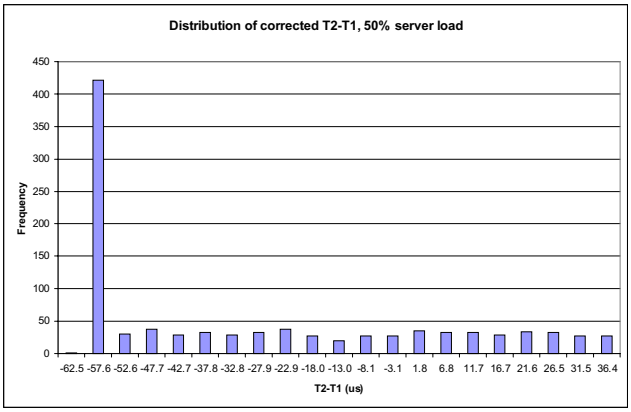

Figure 14 The differences between the values at 50% extra timeserver load

It should be mentioned that estimation of frequency deviation in real life implementations preferably should be a continuous process due to possible temperature variations that might cause fluctuations in the clock frequency. Such a repeated process and the size of the acquisition window are topics for further investigations.

Adjustment of Absolute Time

In the previous section we have shown a method for making the time client and timeserver clocks track each other. What is left is a way to calculate the correction value that should be added to *t1* in order to have two clocks showing the same value at the same point in time.

If we have calibrated the network setup beforehand, we know the theoretical amount of time it takes a time packet to travel from the client to the server. If we add that amount to *t1*, we know what *t1* "ought to have shown" at the global time *t2* and from there it is a simple matter of adjusting an offset value.

At least it is so in the ideal case. The shape of the distribution of the differences from the calculated values and the very small standard deviation shows that using the calculated value for *t2* in the calculations above will put us comfortably within the allowed limits.

Rejecting a Time Stamp

Moving on to the heavily loaded timeserver (Figure 14), the *t2-t1* differences are spread out over a larger area, the standard deviation is large and the average differs a bit from the median. Looking at the figure, we see that getting rid of all the "noise" to the right of the average by filtering, we would have a much more concentrated distribution and a standard deviation that is far below 25µs, meeting the class T3 time requirements. This can be achieved by utilizing the fact that the additional switch delay is caused by another packet being transmitted or being scheduled for transmission when the time packet is put in the output buffer of the switch port connected to the SNTP server. This means that the interval between the current time packet and the previous packet will be very small[8]. If the interval is less than a predefined value, the corresponding time stamp should be rejected.[9].

VII. BEYOND THE SPEED OF LIGHT – CLASS T5

OnTime Networks [17] provides industrial-class fast Ethernet switches fulfilling an extensive list of environment requirements relevant for substation automation applications. The switches are intended for applications with real time requirements and therefore offer some additional features:

- Real time data may be transmitted using packet priority (according to IP Type of Service (ToS) or IEEE 802.1p

8. The interval will be equal to the minimum Interpacket Gap if the switch supports full wire speed (9.6 µs in case of 10BASE and 0.96 µs in case of 100BASE).
9. Patent pending.

and IEEE 802.1Q) in order to guarantee worst-case switch latency.

- Each switch may be delivered with an integrated SNTP timeserver.

The SNTP timeserver in the OnTime switches provides the following:

- Fully compliant with SNTP standard (RFC2030).
- 0.2ppm local clock accuracy.
- Integrated GPS receiver for system clock generation.
- Handle loss of GPS coverage.

An incoming SNTP request packet is time stamped in hardware as soon as the packet enter the switch, and the corresponding SNTP reply packet is sent when actual time is equal to the transmit time stamp given in the SNTP payload. Thus, the traditional problem related to the non-deterministic access to the Ethernet is not a problem here due to the tight interaction between the SNTP timeserver and the switch architecture.

This time synchronization scheme provides the following advantages:

- Timing synchronization accuracy better than one microsecond if time stamping in the time client is performed in hardware, see “Time client implementation issues”.
- Both server time stamps - *t2* (receive) and *t3* (transmit) - may be used at the time client for synchronization purposes, and the drop link propagation delay can easily be removed based on the calculated round trip delay. Today propagation delay is measured manually, while it can be done automatically with this new feature.
- The timing accuracy is independent of the network load.
- No clever filtering/erasure techniques are needed in the time client.
- No patent conflict on the time client side. The time client implementation is entirely based on the SNTP standard (RFC2030), even though time stamping is performed at a low level.
- The Ethernet infrastructure can consist of several switches. Thus, the time accuracy that can be achieved on the time clients does not depend on the complexity of the Ethernet infrastructure as long as there is only one drop-link between a time client and a switch with integrated timeserver.

Figure 15 shows the principles of the OnTime time synchronization scheme. An accuracy better than one microsecond has been measured using an OnTime switch with integrated SNTP timeserver and an ABB time client based on an ARM7TDMI processor. All time stampings were performed in hardware, both on the server and the client side.

The OnTime switch with integrated SNTP timeserver is also a relevant alternative for time clients with less demanding timing accuracy requirements (e.g. class T3, T2 or T1). A time client with class T3 requirements that is directly connected to an OnTime switch with SNTP timeserver will fulfil this requirement even if the time stamping of outgoing and incoming SNTP packets is performed in an Interrupt Service Routine (ISR) outside the RTOS[10]. Class T3 requirements can also be met in case a switch (without timeserver) is installed in-between the OnTime switch with integrated SNTP timeserver and the time client.

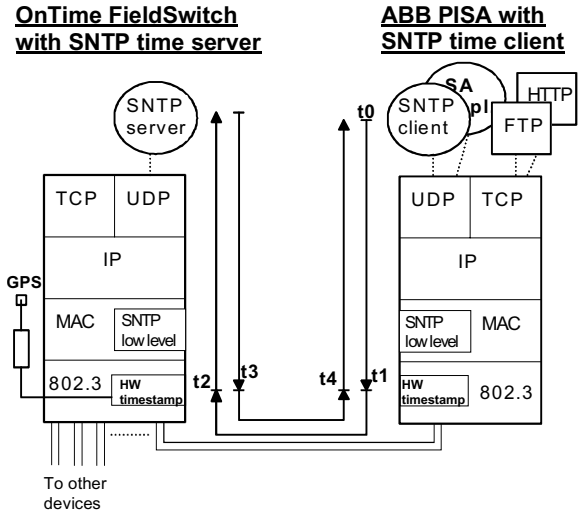

Figure 15 OnTime SNTP time synchronization scheme

VIII. SUMMARY AND CONCLUSIONS

In this paper we have presented general solutions for achieving class T5 (1 µs) and class T3 (25 µs) time synchronization over switched Ethernet. The former is based on an Ethernet switch from OnTime Networks, while the latter relies on standard switches. Common for both solutions is that they adhere to low-level time stamp implementation of the Simple Network Time Protocol (SNTP).

We have shown that by implementing hardware time stamping or low-level software time stamping outside the real time operating system we have eliminated the client inaccuracy from the error budget of the SNTP time synchronization loop. If the SNTP timeserver relies on the same time stamping techniques, the only remaining factor to be handled in the error budget is possible time delay variations within the infrastructure.

For the class T3 solution we have sketched a method for estimating frequency deviation based on statistical calculations and reasoning. In order to reach the class T3 requirement (having an acceptable standard deviation) filtering/erasing of “faulty” SNTP request packets (extra delay caused by head-of-line blocking) must be applied. Such methods exist.

The class T5 requirement is much tougher and can hardly tolerate any jitter within the infrastructure. OnTime Networks provides an elegant solution integrating the SNTP timeserver into industrial-class Fast Ethernet switches. By this integral approach OnTime Networks have solved the non-deterministic Ethernet bus access problems for SNTP request packets,

10. See “*Time Stamping implementation issues*”

making class T5 synchronization over switched Ethernet possible.

IX. REFERENCES

[1] D. W. Allan, J.E. Gray and H.E. Machlan, The National Bureau of Standards atomic time scale: generation, stability, accuracy and accessibility, Blair,B.E. (Ed.), "Time and Frequency Theory and Fundamentals", *National Bureau of Standards Monograph* 140, U.S. Department of Commerce, 205-231, 1974.

[2] H. Attiya, A. Herzberg, and S. Rajsbaum, "Optimal clock synchronization under different delay assumptions", *Proc. 12th Annual ACM Symposium on Principles of Distributed Computing*, 109-120, August 1993

[3] F. Cristian, "A probabilistic approach to distributed clock synchronization", *Distributed Computing* 3, 146-158, 1989.

[4] IEC 61850 Communication Networks and Systems in Substations, Part 5: Communication Requirements for Functions and Device Models CDV Feb. 2001, Part 7-2: Basic Communication Structure for Substations and Feeder Equipment - Abstract Communication Service Interface (ACSI), CDV March 2001..

[5] S. Johannessen, T. Skeie, and C. Brunner, "Ethernet in Substation Automation", submitted to *IEEE Control Systems Magazine*, Oct. 2000.

[6] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems", *IEEE Trans. Computers* C-36, 933-939, 1987.

[7] C. LeBlanc, "The Future of Industrial Networking and Connectivity", *Dedicated Systems Magazine*, March 2000.

[8] D.L. Mills, "Internet time synchronization: the Network Time Protocol", *IEEE Trans. Communications* COM-39, 1482-1493, 1992.

[9] D.L. Mills, "Precision synchronization of computer network clocks", *ACM Computer Communication Review* 24, 28-43, 1993.

[10] D.L. Mills, "Improved algorithms for synchronizing computer network clocks", *IEEE/ACM Trans. On Network*s, 245-254, 1995.

[11] D.L. Mills, "Adaptive hybrid clock discipline algorithm for the Network Time Protocol", *IEEE/ACM Trans. On Networking* 6, 505-514, 1998.

[12] D.L. Mills, "Simple Network Time Protocol", *RFC 2030*, University of Delaware.

[13] J. Tengdin, M.S. Simon, and C.R. Sufana, LAN Congestion Scenario and Performance Evaluation, Proceedings IEEE Power Engineering Society, Winter Meeting 1999.

[14] 3COM SuperStack II Switch 3300, http://www.3com.com/products/switches/index.html

[15] BayStack 450-24T switch, Nortel, http://www.nortelnetworks.com/products/switching/

[16] CISCO Catalyst 3524XL switch, http://www.cisco.com/warp/public/44/jump/switches.shtml

[17] OnTime Networks, http://www.ontimenet.com/