PAPER Special Section on the Next Generation Ethernet Technologies

Hardware-Based Precise Time Synchronization on Gb/s Ethernet Enhanced with Preemptive Priority

Yoshiaki YAMADA[†a)], **Satoru OHTA**[†b)], ***and*** **Hitoshi UEMATSU**[†c)], ***Members***

SUMMARY Time synchronization is indispensable for wide area distributed systems including sensor networks, automation systems, and measurement/control systems. Another application is clock distribution, which is indispensable to support continuous information transfer. Because of the increasing demand for more sophisticated applications, it is essential to establish a time synchronization technique that offers higher accuracy and reliability. Particularly, the accuracy of time synchronization for Ethernet must be enhanced since Ethernet is becoming more important in telecommunication networks. This paper investigates a precise time synchronization technique that supports Gb/s Ethernet. To obtain accurate time synchronization, delay variation in message transfer and processing must be minimized. For this purpose, the paper first describes the implementation of preemptive priority queuing, which decreases the message delay variation of Ethernet. Through experiments, it is shown that preemptive priority queuing effectively achieves very low delay variation. The paper then proposes a method to synchronize the time signal of a slave node to that of the master node. The proposed time synchronization method is performed in the lower protocol layer and implemented on FPGA-based hardware. The method achieves superior time accuracy through the low message transfer/processing delay variation provided by preemptive priority, lower layer execution, and hardware implementation. The effectiveness of the method is confirmed through experiments. The experiments show that the time variation achieved by the method is smaller than 0.1 μsec. This performance is better than those obtained by existing synchronization methods.

key words: *Ethernet, synchronization, delay variation, priority, QoS*

1. Introduction

Time synchronization [1] is indispensable for wide area distributed systems including sensor networks [2], automation systems [3], [4] and measurement/control systems [5], [6]. Another application is clock distribution, which is indispensable to support continuous information transfer. For clock distribution, the employment of time synchronization is advantageous because it greatly improves the phase accuracy of the clock signal [7]. Because of the increasing demand for more sophisticated applications, it is essential to establish a time synchronization technique that offers higher accuracy and reliability.

Of particular interest is time synchronization for Ethernet. This is because Ethernet is the most popular data link protocol and covers applications from LANs (Local Area Networks) to WANs (Wide Area Networks). Namely, Ethernet time synchronization would enhance various types of networks. Additionally, because of the popularity of Ethernet, its components are inexpensive. Thus, Ethernet time synchronization would also be beneficial from the cost viewpoint.

Time can be synchronized on Ethernet by employing the NTP (Network Time Protocol [1], [8]) or its subset SNTP (Simple Network Time Protocol [9]), both of which run on IP (Internet Protocol). The drawback of NTP and SNTP is accuracy. The reported time accuracy of NTP is 1–50 msec [8], [12], [13] and this is insufficient for some applications. The accuracy of (S)NTP can be greatly improved by executing time-stamping in hardware. It was reported that this approach yields the accuracy of several tens of microseconds [3]. One alternative, IEEE 1588 [5], was created for measurement and control on small networks. An implementation of IEEE 1588 achieves the accuracy of several hundred nanoseconds [6]. Another method is GPS (Ground Position System). However, GPS is not suited to be installed on many nodes in a network because of its antenna placement limitation.

The time accuracy possible with standard Ethernet is hampered by the tradition of implementing most functions as software processes. For example, message transfer/processing delay is often varied by the software implementation of packet multiplexing. Reports on precise time synchronization [3], [6] rely on the standard Ethernet and the processing is mostly executed in software. Adequate accuracy can only be achieved by modifying the Ethernet protocol to decrease message delay variation implementing more functions on hardware.

This paper investigates a more accurate time synchronization technique for Gb/s Ethernet. Its application area includes clock signal distribution for synchronous networks. For this purpose, existing methods are not sufficiently accurate and thus they fail to well handle jitter and wander (See [10] and [11] for jitter and wander requirements). To obtain higher timing precision, this paper first shows an implementation of preemptive priority queuing, which minimizes the message transfer delay variation. It then proposes a time synchronization method, which is executed in the lower protocol layer and implemented on FPGA (Field Programmable Gate Array) hardware. The proposed synchronization method achieves superior accuracy due to the smaller message delay variation provided by the preemptive priority technique. The lower layer execution and the hardware-based implementation further suppress the inaccuracy induced by processing delay variation. Its effectiveness is confirmed by an experiment.

Manuscript received July 11, 2005.

[†]The authors are with the NTT Network Innovation Laboratories, NTT Corporation, Yokosuka-shi, 239-0847 Japan.

a) E-mail: yamada.yoshiaki@lab.ntt.co.jp

b) E-mail: ohta.satoru@lab.ntt.co.jp

c) E-mail: uematsu.hitoshi@lab.ntt.co.jp

DOI: 10.1093/ietcom/e89–b.3.683

2. Related Works

Currently, the most popular time synchronization method is to employ NTP (or SNTP). Since NTP (or SNTP) works on the IP protocol stack, it is easily adapted to Ethernet. However, its accuracy in terms of synchronization is not good enough. With NTP, the time of the client host is synchronized to that of the server by adjusting the client clock so as to cancel the time difference between the client and the server. The time difference is measured by exchanging messages between the host and the server. In this process, the measured time difference is disturbed by variation in the message transfer delay. This disturbance degrades the time accuracy. Delay variation is unavoidable because timing messages collided with other messages and are queued in the buffers of switches or routers in the network. Moreover, NTP is an application layer protocol, and is usually implemented as software. Thus, collision among application processes also triggers delay variation and further degrades time accuracy. Thus, it is unrealistic to expect that NTP can achieve highly accurate synchronization.

It has been reported that the time synchronized by NTP exhibits deviations of 28.7 msec for a WAN [12] and 1 msec for LANs [13]. This performance is not sufficient for some applications.

More precise time synchronization methods for Ethernet were reported in [3], [4]. These methods, which basically employ SNTP, successfully improve time accuracy by the hardware-based timestamp approach. It was reported that they can achieve accuracy ranging from 25 μsec to 1 μsec. However, since these methods rely on software processing for the most part, other than timestamp handling, they can not avoid the accuracy degradation caused by conflicts among software processes. Actually, the measured accuracy achieved varies with the load on the server [3].

One time synchronization scheme designed for measurement and control was standardized as IEEE 1588 [5]. The protocol specified in IEEE 1588 supports precise time synchronization by measuring the time of message departure on the physical media. It was reported that the mean value of time variation achieved by an IEEE 1588 implementation was 49 nsec and the standard deviation was 233 nsec [6]. The drawback of IEEE 1588 is that its accuracy is not guaranteed if the messages pass through routers. Thus, the method should be used only for synchronization within a subnet. It is possible to obtain better performance than IEEE 1588 by implementing time synchronization in the lower protocol layer.

For precise time distribution, GPS is also an effective technique. It offers a time accuracy of 10 nsec [14]. Unfortunately, it requires that a antenna sited so as to always have a clear view of one or more of the GPS satellites. Because of this limitation on antenna placement, it is difficult to employ GPS-based time distribution in every node in a network.

3. Preemptive Priority Queuing

This study employs preemptive priority queuing to minimize the delay and delay variation. Preemptive and nonpreemptive are the two basic approaches to priority queuing [15]. With preemptive priority queuing, if the system receives a high-priority frame while processing a low-priority frame, it stops processing the latter and begins processing the former. The system resumes processing the low priority frame after the high-priority frame is processed. In this scheme, high-priority frames are processed immediately upon arrival so they experience smaller delay and delay variation.

The basic approach of Ethernet is nonpreemptive queuing. To implement preemptive priority queuing, the receiver must identify and extract high-priority Ethernet frames as well as reconstructing the low priority frame. Reference [16] reports an implementation of such functions. With the method of [16], the sender inserts special codes before and after each high priority frame. The special codes are bit sequences that are unused by the standard Ethernet protocol. The receiver uses the sequences to identify the position of the high priority frames and extract them. This process is illustrated in Fig. 1.

This study used a commercial Gb/s Ethernet multiplexer [17] to implement preemptive priority queuing in a manner similar to that used in [16]. While Ref. [16] placed the preemptive priority mechanism in the PHY layer, this study placed it between the MAC and PHY layers as shown in Fig. 2. We implemented the priority queuing mechanism using FPGAs. Note that we did not modify the PHY layer. Thus, generic and inexpensive PHY layer components (i.e. ICs) can be employed. This functional arrangement is expected to be more economical than that of Ref. [16]. In addition, this study applies cut-through [18] to high-priority traffic to decrease the delay further.

Our implemented version of preemptive priority queuing was evaluated by an experiment using the configuration shown in Fig. 3. As the figure shows, two multiplexers with preemptive priority queues were connected by Gb/s

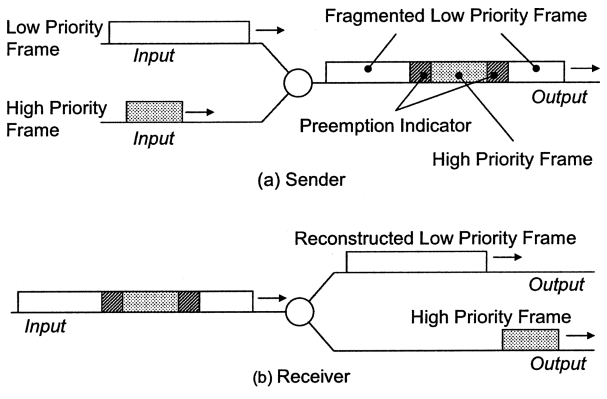


Fig. 1 Preemptive priority control executed at (a) the sender and (b) the receiver, connected by Ethernet.

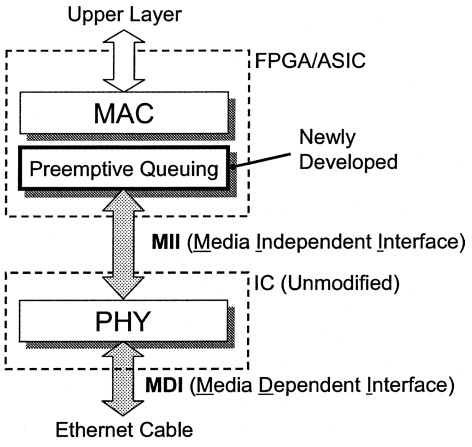


Fig. 2 Layer diagram for the implementation of preemptive priority queuing.

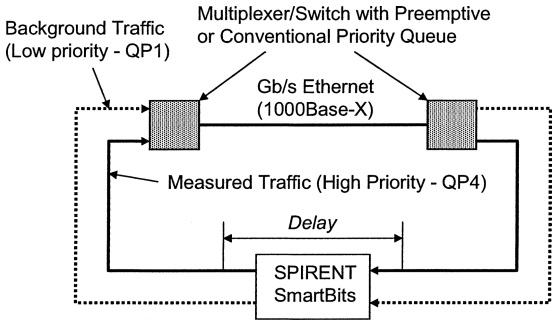


Fig. 3 Measurement configuration to evaluate preemptive priority queuing.

Ethernet. High priority traffic was transmitted from one multiplexer to the other and the average delay of Ethernet frames was measured. Simultaneously, low priority frames were also transferred as background traffic. The priority was indicated by the priority tag specified in IEEE 802.1p. Namely, high priority traffic was tagged QP4, while low priority background traffic was tagged QP1. The frame length of the measured traffic was 1518 bytes, and its inter-arrival time was constant. Meanwhile, the inter-arrival time of the background traffic was also constant, but its frame length was random. The bit rate of the background traffic was set at 20%, 50%, and 80% of the wire speed, 1 Gb/s. For comparison, the same measurements were executed using commercial switches (Summit 1i, Extreme Networks [19]) in place of the multiplexers. The switches use non-preemptive priority queuing. SmartBits (SPIRENT [20]) was used for traffic generation and delay measurements.

Figure 4 shows the results. In Fig. 4, the x-axis is the bit rate of the measured traffic, while the y-axis is the average delay. The bit rate is shown as a percentage of the wire speed. The figure shows that the preemptive priority traffic had delay of less than 10 μsec while the non-preemptive priority architecture suffered delay of more than 30 μsec. Moreover, this increased to 50–60 μsec when the total bit rate of the measured and background traffic exceeded 100%

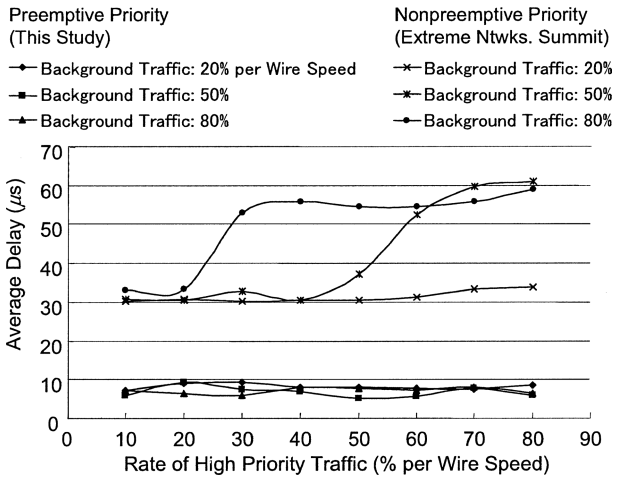


Fig. 4 Delay characteristics of preemptive priority queuing and conventional queuing (no preemption).

of the wire speed. By contrast, the delay of the preemptive priority traffic was almost regardless of the traffic load. That is, the performance of the high priority traffic was independent of the background traffic. This characteristic suggests that preemptive priority queuing prevents the high priority frames from contending with the low priority frames for transmission resources. Since the delay variation is brought by frame contention, it is expected that preemptive queuing yields very low delay variation for high priority frames. By utilizing this characteristic, the accuracy of time synchronization can be greatly improved.

Figure 4 also suggests that non-preemptive priority produces substantial time offsets if the traffic is strongly asymmetric. Suppose that heavy background traffic runs in one direction. The delay in that direction can become 20 μsec larger than that in the reverse direction. This is a serious problem since popular synchronization methods (NTP or IEEE 1588) assume that the delay is the same in both directions. These methods allow the delay difference to create time offset. Preemptive priority, by contrast, holds the delay the same in both direction because delay does not depend on the background load. This characteristic effectively avoids the time offsets brought by delay difference between the directions.

4. Time Synchronization

This section proposes a highly accurate time synchronization method. The proposed method employs relatively simple logic that is easily implemented on hardware; it avoids the processing delay variation and performance bottleneck inherent in the software implementations used by the conventional methods. Another feature is that it works on a lower protocol layer. This feature also eliminates the delay variation caused by the upper layer protocol processes and leads to better accuracy.

The proposed method was realized on two nodes connected by Gb/s Ethernet with preemptive priority queues as

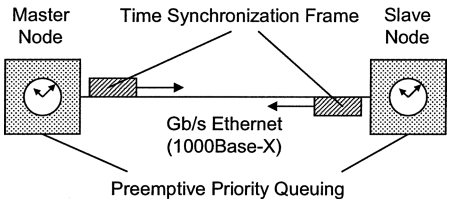


Fig. 5 Basic configuration for the proposed time synchronization method.

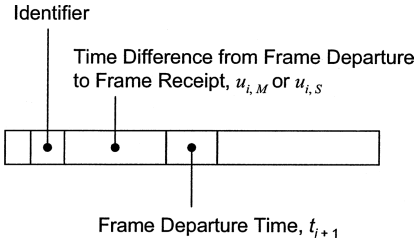


Fig. 6 Structure of time synchronization frame.

shown in Fig. 5. In the following, the system shown in Fig. 5 is referred to the "basic configuration." In the basic configuration, one node acts as the "master" while the other node acts as the "slave."

Since the proposed method performs in the lower protocol layer (below the MAC layer), it does not work between two nodes connected via routers. Nevertheless, the method can be applied to a network with more than two nodes. Namely, in such a network, we can discover a tree, whose root is the time source. The basic configuration is then applied to each edge of the tree such that the node closer to the root is the master. The node clocks are then synchronized to the clock of the root following the paths from the root to the leaves.

The proposed method synchronizes the slave clock to the master clock by periodically exchanging synchronization frames. As shown in Fig. 6, the synchronization frame has three fields. The first field is the identifier. This field indicates that the frame type is "synchronization frame." The second field carries a timestamp that shows its departure time. The third field is the time difference between frame departure and frame receipt. Each synchronization frame is transmitted as high priority traffic using preemptive priority queuing and thus suffers very slight delay variation. By exchanging synchronization frames, the slave node can find the time offset between its local (slave) clock and the master clock. Canceling the time offset synchronizes the local clock to the master clock.

4.1 Time Offset Evaluation

In the basic configuration, the slave node needs to find the time offset between its local clock and the master clock. The time offset is computed as follows.

Let Δ denote the time offset. If the time indicated on the slave clock is t, the time indicated on the master clock is $t + \Delta$; d denotes the one-way delay between the slave node and the master node. Note that d does not depend on the direction because media length is the same in both directions and the queuing delay is basically independent of direction.

The master and slave nodes transmit synchronization frames when their local clocks indicate $t_0, t_1, t_2, \ldots$. If the slave node sends a synchronization frame at time t_i on the slave clock, its departure time is $t_i + \Delta$ on the master clock. If this frame arrives the master node at time $t_{i,r,M}$ on the master clock, then,

$$t_{i,r,M} = t_i + d + \Delta. \tag{1}$$

Similarly, if the master node sends a synchronization frame at t_i, its departure time is $t_i - \Delta$ on the slave clock. If this frame arrives the slave node at time $t_{i,r,S}$ on the slave clock, then,

$$t_{i,r,S} = t_i + d - \Delta. \tag{2}$$

Each node computes the time difference between the departure and receipt of the just received synchronization frame. For the frames sent at time t_i, let $u_{i,M}$ denote the time difference computed at the master node, and let $u_{i,S}$ denote that computed at the slave node. Then, the following equations are obtained from Eqs. (1) and (2).

$$u_{i,M} = d + \Delta. \tag{3}$$

$$u_{i,S} = d - \Delta. \tag{4}$$

Time differences $u_{i,M}$ is written to the time difference field (Fig. 6) of the next synchronization frame, which leaves the master node at time t_j ($j > i$). By receiving this frame, the slave node can recognize values $u_{i,M}$ and $u_{i,S}$. Δ and d are then computed at the slave node using the following equations.

$$\Delta = (u_{i,M} - u_{i,S})/2. \tag{5}$$

$$d = (u_{i,M} + u_{i,S})/2. \tag{6}$$

Time is synchronized by advancing or retarding the slave clock to cancel Δ derived by Eq. (5). Namely, the clock should be advanced for $\Delta > 0$ or retarded for $\Delta < 0$. In the synchronization frame, the frame transmission time t_i should be indicated in the timestamp field (Fig. 6) to compute Δ correctly even if Δ is larger than the transmission interval $t_{i+1} - t_i$. Figure 7 illustrates the above procedure of computing time offset Δ. Although one-way delay d obtained by Eq. (6) is not used for synchronization, it will be useful for network management.

The proposed method of evaluating time offset uses fewer parameters than NTP. This feature is obtained by transmitting synchronization frames at set times, $t_0, t_1, \ldots$. The decrease in parameters is desirable because it simplifies hardware implementation. Additionally, since fewer parameters means shorter message length, it is also advantageous with regard to bandwidth consumption.

4.2 Time Signal and Its Control

The time synchronization mechanism requires an appropriate time signal that is controllable by time offset. Such a

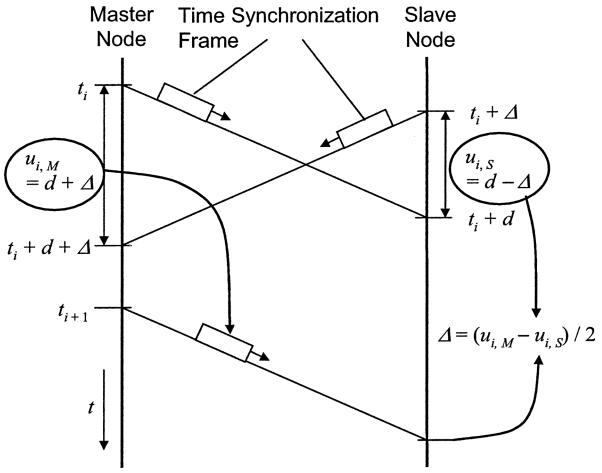


Fig. 7 Principle of measuring time difference between the master and slave nodes.

signal is obtained in the following way.

The proposed method sets the interval between two successive synchronization frames to a constant value, denoted by T. If time offset is 0, the master and the slave have identical synchronization frame departure times. This means that the synchronization frame departure time is usable as the synchronized time signal.

The time signal defined above is easily controlled to cancel time offset by employing two types of synchronization frames. The difference between these types is the bit length. One frame type has a shorter bit length, denoted by y_1 bits, while the length of the other frame is denoted by y_2 ($y_1 < y_2$) bits. After a node sends a synchronization frame, it waits for the time specified for sending x bits. It then sends the next synchronization frame. During this process, if the short synchronization frame is used, the frame interval decreases and thus the time signal is advanced. Conversely, if the long frame is used, the time signal is retarded. Therefore, the time signal can be adjusted through frame type selection depending on the time offset.

If the time offset is 0 or if the node is the master, the two frame types are transmitted alternately. The period length of the time signal remains synchronized even if the bit rates differ between the slave and the master. If α is the ratio of the short frames among the synchronization frames, the average interval between synchronization frames is expressed as follows.

$$T = \{x + \alpha y_1 + (1 - \alpha) y_2\} / r, \tag{7}$$

where r is the bit rate on the transmission media. The proposed control automatically sets value α such that the master and the slave have the same T value.

In the experiment described in the next section, the parameters are $x = 124800$ bits, $y_1 = 192$ bits, $y_2 = 208$ bits and $r = 1$ Gb/s. This setting yields $T = 125\,\mu$sec. Figure 8 explains the control process of the time signal.

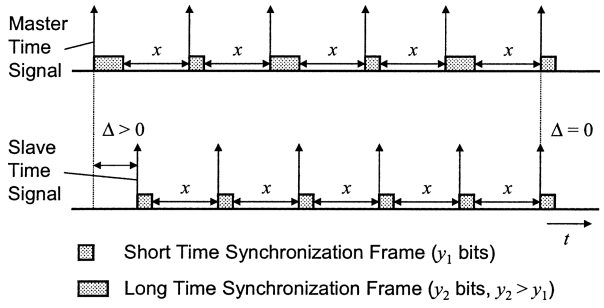


Fig. 8 Time signal generated by the transmission of synchronization frames.

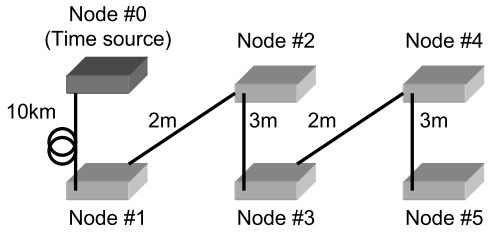


Fig. 9 Experimental network for the evaluation of the proposed time synchronization scheme.

5. Experiment

The proposed synchronization method (Sect. 4) was implemented with preemptive priority queuing (Sect. 3) on an FPGA. The synchronization method was placed also placed between the MAC and PHY layers for preemptive priority queuing. This hardware implementation was used in an experiment to confirm the effectiveness of the proposed method. The network used in the experiment consists of six serially connected nodes as shown in Fig. 9. In this configuration, node #0 works as the time source. By applying the basic master/slave configuration to the two directly connected nodes, the clocks of nodes #1, #2, ..., #5 are synchronized to that of node #0. Time synchronization was examined by comparing the wave forms of the time signals on an oscilloscope.

Figure 10 shows the results gathered. The upper side of the figure compares the time signals of nodes #0, #3 and #5. The time signal period is 125 μsec. As shown in the figure, the signals of nodes #3 and #5 are completely synchronized to the signal of node #0.

The lower side of Fig. 10 shows enlargements of the wave forms on the time axis. From the figure, the time signals of nodes #3 and #5 vary slightly. The time variation, however, is smaller than 0.1 μsec. Previous implementations of precise time synchronization on Ethernet allowed the time variation to range from several tens of milliseconds [3] to several hundreds of microseconds [6]. Therefore, the proposed method achieves much better accuracy than the existing methods. This is due to the use of preemptive priority and hardware processing, both of which greatly reduce message/processing delay variation.

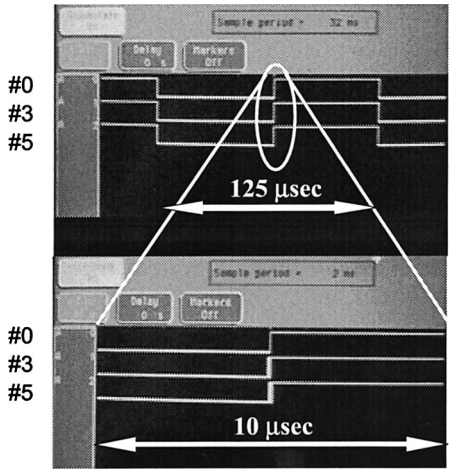


Fig. 10 Measurement results of the synchronized timing signal.

In the lower side of Fig. 10, no significant difference is seen between the time variations of nodes #3 and #5. This implies that the proposed method achieves highly accurate synchronization even if the basic configuration is concatenated many times in a large network. This characteristic is suitable to the clock signal distribution application, which requires accuracy as well as scalability for a large sized network.

6. Conclusion

This paper presented a precise time synchronization method for Gb/s Ethernet. First, the paper described preemptive priority queuing as implemented on Ethernet. This queuing scheme effectively decreases the frame transfer delay and its variation, and so enhances time accuracy. Next, the paper proposed a time synchronization method that works on a lower layer of Ethernet and is well suited to hardware implementation. The method utilizes preemptive priority queuing to obtain better time accuracy. The accuracy is further improved by our hardware-based implementation. An experiment showed that the proposed synchronization method achieves superior time accuracy over existing methods. In addition, time was accurately synchronized even if the basic master-slave configuration was repeatedly applied for several serially connected nodes. These results imply that the proposed mechanism can provide precise clock signal over a large network successfully.

Acknowledgments

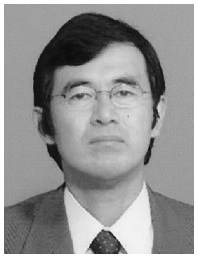
The authors would like to thank Dr. Masahiro Morikura for his support and encouragement.

References

[1] D. Mills, "Internet time synchronization: The network time protocol," IEEE Trans. Commun., vol.39, no.10, pp.1482–1493, Oct. 1991.
[2] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: A survey," IEEE Netw., vol.18, no.4, pp.45–50, July/Aug. 2004.
[3] T. Skeie, S. Johannessen, and Ø. Holmeide, "Highly accurate time synchronization over switched Ethernet," Proc. 8th IEEE Conference on Emerging Technologies and Factory Automation (EFTA '01), pp.195–204, Antibes, France, Oct. 2001.
[4] OnTime Networks, http://www.ontimenet.com/
[5] IEEE Standard 1588, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, Nov. 2002.
[6] J. Eidson and K. Lee, "Sharing a common sense of time," IEEE Instrum. Meas. Mag., vol.6, no.1, pp.26–32, March 2003.
[7] A. Imaoka and M. Kihara, "Time signal distribution in communication networks based on Synchronous Digital Hierarchy," IEEE Trans. Commun., vol.45, no.2, pp.247–253, Feb. 1997.
[8] D. Mills, Network Time Protocol (Version 3) specification, implementation and analysis, IETF RFC1305, March 1992.
[9] D. Mills, Simple Network Time Protocol (SNTP) version 4 for IPv4, IPv6 and OSI, IETF RFC2030, Oct. 1996.
[10] ITU-T Recommendation G.812, Timing Requirements of Slave Clocks Suitable for Use as Node Clocks in Synchronization Networks, June 1998.
[11] ITU-T Recommendation G.824, The Control of Jitter and Wander within Digital Networks Which Are Based on the 1544 kbit/s Hierarchy, March 2000.
[12] D. Mills, A. Thyagarjan, and B. Huffman, "Internet timekeeping around the globe," Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting, pp.365–371, Long Beach, CA, Dec. 1997.
[13] D. Mills, "Precision synchronization of computer network clocks," ACM SIGCOM Comput. Commun. Rev., vol.24, no.2, pp.28–43, April 1994.
[14] W. Lewandowski, G. Petit, and C. Thomas, "Precision and accuracy of GPS time transfer," IEEE Trans. Instrum. Meas., vol.42, no.2, pp.474–479, April 1993.
[15] L. Kleinrock, Queuing Systems—Vol.II: Computer Applications, John Wiley & Sons, 1976.
[16] S. Yanagimachi, T. Yoshikawa, T. Baba, and K. Fukuchi, "An architecture of CIAO (Cut-In-Any Occasion) which makes a virtual private channel in a PHY layer of the Ethernet," IEICE Technical Report, CS2003-105, Nov. 2003.
[17] NTT Electronics, http://www.nel.co.jp/product/optical/gigabee2.html
[18] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," Comput. Netw., vol.3, pp.267–286, 1979.
[19] Extreme Networks, http://www.extremenetworks.com/
[20] SPIRENT Communications, http://www.spirentcom.com/

Yoshiaki Yamada received the B.E. and M.E. degrees from the University of Electro-Communications, Tokyo, Japan, in 1990 and 1992, respectively. Since 1992, he has been with NTT Corporation, working on the research and development of optical switching and optical burst communications. He is currently a senior research engineer at NTT Network Innovation Laboratories, NTT Corporation. His research interest includes the time synchronization technique and high-speed layer 2 protocols. He is a member of the IEEE. He received the Young Researchers' Award from the IEICE in 1998.

Satoru Ohta received the B.E., M.E., and Dr. Eng. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 1981, 1983, and 1996, respectively. Since 1983, he has been with NTT, where he worked on the research and development of cross-connect systems, broadband ISDN, network management, and telecommunication network planning. He is currently involved in research on network protocols and network performance evaluation at NTT Network Innovation Laboratories, NTT Corporation. Dr. Ohta is a member of the IEEE. He received the Excellent Paper Award from the IEICE in 1991.



Hitoshi Uematsu was born in Hyogo, Japan, on July 1, 1960. He received the B.S., M.S. and Ph.D. degrees in communication engineering from Osaka University in 1983, 1985, and 1993, respectively. He joined the NTT Laboratories, Yokosuka, Kanagawa, Japan, in 1985. From 1985 to 1995, he was engaged in research on ATM (asynchronous transfer mode) cross-connect switches. He also researched circuit signal transfer systems and their clock transfer techniques by the SRTS (synchronous residual time stamp) method in AAL (ATM adaptation layer) type 1. From 1996 to 1998, he was engaged in the development of virtual path based ATM leased line systems, named "ATM Mega-Link service" and ATM-PON (passive optical network) access systems. In 1999, he joined NTT Network Innovation Laboratories and he engaged in research on optical ADM transport systems based on wavelength multiplexing techniques. Since 2002, he has been engaged in Ethernet voice transfer techniques with priority control and a digital RoF (radio on fiber) system. Dr. Uematsu is a member of the IEEE.