TELKOMNIKA, Vol. 11, No. 7, July 2013, pp. 3826 ~ 3833
e-ISSN: 2087-278X

Precise Clock Synchronization Algorithm for Distributed Control Systems

Yi Liu*[1], Haiying Yang[2]
[1,2]Department of Computer Center, Liaoning University of Technology
Jinzhou, 121001, PR China
*Corresponding author, e-mail: jzly71@126.com[*1], yang_haiying@163.com[2]

Abstract

The distributed control systems are applied widely in communication, navigation and power fields. With growing in both systems complexity and speed of data exchange, the real time performance of systems is required. Existing clock synchronization techniques include those for wide-area network (WAN) applications with large propagation delays, e.g., the Network Time Protocol (NTP), and those for control systems over local-area networks (LANs), e.g., the IEEE 1588 Precise Time Protocol (PTP). However, NTP-like protocols have only millisecond-scale precision, which is too coarse for many LAN applications such as instrument monitoring systems, high-quality digital audio systems and sensor networks. The 1588 PTP like protocols, which are still under development, require support of a highly precise hardware clock. The key point of the existing Ethernet test and measurement is that the method needs to achieve precision clock synchronization between different terminals. In this paper, an improved method is proposed with consideration of the time drift and propagation delay, expressing master-slave synchronization as a linear relationship and inducting the line of least squares fitting algorithm that traded as linear matrix form. The precision time protocol (PTP) is also analyzed in this paper. The experiment results show that the synchronization accuracy reaches sub-microsecond grade, which is the same as the PTP.

Keywords*: distributed control systems, clock synchronization algorithm, time drift*

1. Introduction

With the emergence of new time-critical distributed computing paradigms, such as distributed control converters, human-robot physical interaction and ad-hoc networks with a large number of disposable sensors, precise clock synchronization, which is an old topic of research, has regained increasing attention as a critical issue in distributed computing environments [1-5]. To achieve clock synchronization, network devices exchange synchronization messages, which carry timestamps, for time computation and error compensation. Therefore, the field of clock synchronization focuses on the design of message exchanges, offset computations, and compensation for clock drifts.

Clock synchronization refers that each node clock in the networked and the various application system clock connecting to the networked is synchronous with the Coordinated Universal Time (Coordinated Universal Time UTC). With the increasing of the clock synchronization requirements in networked control system, the clock synchronization in distributed systems, which are based on Ethernet, requires precision to the microsecond grade. Therefore, the clock synchronization algorithm becomes the research hotspot.

Currently, there are three algorithms: Networked Time Protocol (NTP), Simple Networked Time Protocol (SNTP) and IEEE 1588. Mills [6] proposed the networked time protocol (NTP) service to handle large networkeds (Internet) with large and variable message delays. In the WAN, the time accuracy provided by NTP is tens of milliseconds, and sub-millisecond in the LAN. As the simplification of NTP, SNTP applies to the situation in which it not necessary to use NTP completely. John Edson [7] proposed IEEE1588 service to Ethernet or distributed bus systems using multicast mode. The time accuracy of IEEE 1588 is sub-microsecond. At present, most of researchers have focused on the use of hardware and software technologies to achieve the clock synchronization, and pay less attention to the theoretical research, so, the paper is mainly theoretical derivation of synchronization algorithm.

Received September 17, 2012; Revised January 10, 2013; Accepted February 19, 2013

This paper is organized as follows. The description of the Algorithm is described in Section 2. Results and discussions are presented in section 3. Finally, Section 4 concludes the paper.

2. Research Method

2.1. Description of the Algorithm

The reference time in subnet is the clock of control server node and on this basis, a control client node initiates the clock synchronization to readjust its clock.

Parameter Definition. Clock skew (δ) is defined as the instantaneous difference between the recorded time and the actual absolute time. Clock drift (ρ) occurs in a crystal-based clock, if the counting frequency varies, causing a timing error in the particular clock and generates a diverging clock skew. Let Cc denotes the control client node requesting the clock time. Cs denotes the responding control server node. N node exist in networked, let $t_a^{c_c}(n)|_{n=1}^N$ be the series of time-stamps when the control client sent the clock request, $t_b^{c_s}(n)|_{n=1}^N$ be the series of time-stamps when the control server received and immediately replied to the clock request, and $t_c^{c_c}(n)|_{n=1}^N$ be the series of time-stamps when the response to the clock request arrived back at the local client. Defining t_{abs} as the absolute time, $t_i^{c_c}$ as the local time of node Cc, $t_i^{c_s}$ as the local time of node Cs, $t_n^{c_c c_s}$ and $t_n^{c_s c_c}$ as the transmission times between the control server and the control client.

Algorithm Derivation. Revise the clock drift between the server and the client. Assuming both clocks in the control client and the server has a linear growing skew [8] to the absolute time:

$$t_i^{c_c} = t_{abs} + \delta^{c_c} + \rho^{c_c} t_{abs} \quad (1)$$

$$t_i^{c_s} = t_{abs} + \delta^{c_s} + \rho^{c_s} t_{abs} \quad (2)$$

Combining (1) and (2) to eliminate t_{abs}, one obtains the following linear relation:

$$t_i^{c_s} = k t_i^{c_c} + c \quad (3)$$

In which

$$c = \delta^{c_s} - (\frac{1+\rho^{c_s}}{1+\rho^{c_c}})\delta^{c_c} \quad (4)$$

$$k = \frac{1+\rho^{c_s}}{1+\rho^{c_c}} \quad (5)$$

Delay measurement. It is common to assume that $t_n^{c_c c_s}$ is equal to $t_n^{c_s c_c}$: $E[t_n^{c_c c_s}] = E[t_n^{c_s c_c}] = \mu \quad n \in [1,2,\ldots,N]$

$$\hat{\mu} = \frac{1}{2N}\sum_{n=1}^{N}(t_c^{c_c}(n) - t_a^{c_c}(n)) \quad (6)$$

The data points in the time series $t_b^{c_s}(n)|_{n=1}^N$ and $t_b^{c_c}(n)|_{n=1}^N$ can be written as:

$$t_b^{c_s}(n) = k(t_a^{c_c}(n) + t_n^{c_c c_s}) + c \quad (7)$$

$$t_c^{c_c}(n) = t_a^{c_c}(n) + t_n^{c_c c_s} + t_n^{c_s c_c} \tag{8}$$

Respectively, from equation (7), we obtain:

$$t_b^{c_s}(n) = k(t_a^{c_c}(n) + \hat{\mu}) + c \tag{9}$$

Hence, by fitting a straight line through the data points of plot $t_b^{c_s}(n)|_{n=1}^{N}$ versus $t_a^{c_c}(n)|_{n=1}^{N} + \hat{\mu}$, the gradient $\hat{k}$ and intercept $\hat{c}$ can be determined, which are estimate for k and c, respectively. Here, the least squares line-fitting algorithm [9] is employed which is given as:

$$\begin{bmatrix} \hat{c} \\ \hat{k} \end{bmatrix} = \left[\left(T_a^{c_c}\right)^T T_a^{c_c} \right]^{-1} \left[\left(T_a^{c_c}\right)^T \left(t_b^{c_s}\right) \right] \tag{10}$$

Where

$$T_a^{c_c} = \begin{bmatrix} 1 & t_a^{c_c}(1) + \hat{\mu} \\ 1 & t_a^{c_c}(2) + \hat{\mu} \\ \cdots & \cdots\cdots \\ 1 & t_a^{c_c}(N) + \hat{\mu} \end{bmatrix} \tag{11}$$

And

$$t_b^{c_s}(n) = \left[t_b^{c_s}(1) \quad t_b^{c_s}(2) \quad \ldots\ldots \quad t_b^{c_s}(N) \right]^T \tag{12}$$

Consequently, using the estimate of k, c and μ from (6) and (10), the clock reading in the control server can be transformed to the clock reading in the control client through (3).

Determine Whether the System is Synchronized. Clock synchronization in real-time systems is an ongoing process. In order to avoid frequent repetition of clock synchronization, we need to compute the value of server-client clock offset. After the server clock synchronizes with the client clock, the client clock will calculate each clock synchronization packet that released from the server clock, if the calculated offset value is not less than or not equal to a very small number which determined in advance , it shows the two clocks are in different status and need to be online regulated.

Clock offset (Δ) is defined as $\delta^{c_c} - \delta^{c_s}$.

$$t_n^{c_c c_s} = t_b^{c_c}(n) - t_a^{c_c}(n) = (t_b^{c_s}(n) - \Delta) - t_a^{c_c}(n)$$

$$t_n^{c_s c_c} = t_c^{c_c}(n) - t_b^{c_c}(n) = t_c^{c_c}(n) - (t_b^{c_s}(n) - \Delta)$$

It is common to assume that $E[t_n^{c_c c_s} - t_n^{c_s c_c}] = 0$

$$\Delta = E\left[\frac{2t_b^{c_s}(n) - t_a^{c_c}(n) - t_c^{c_c}}{2} \right] \tag{13}$$

2.2. Synchronization of The IEEE 1588

Using the precision time protocol (PTP) which is defined by IEEE 1588, the whole networked clock is divided into two types, which are named the master clock and the slave clock. The optional clock of the whole system is the grandmaster clock, the master clock within each PTP is selected by the best master clock algorithm [10, 11]. Figure 1 shows a typical master-slave clock configuration diagram.

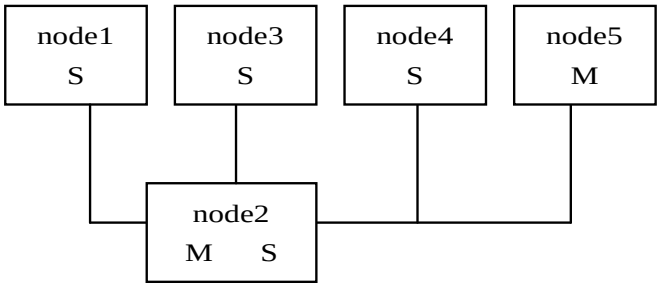

Figure 1. Master-slave configuration

Figure 2 shows the synchronization principle. Moreover, we can see from Figure 2 that the synchronization process is divided into two parts that are separately named the offset measurement and the delay measurement.

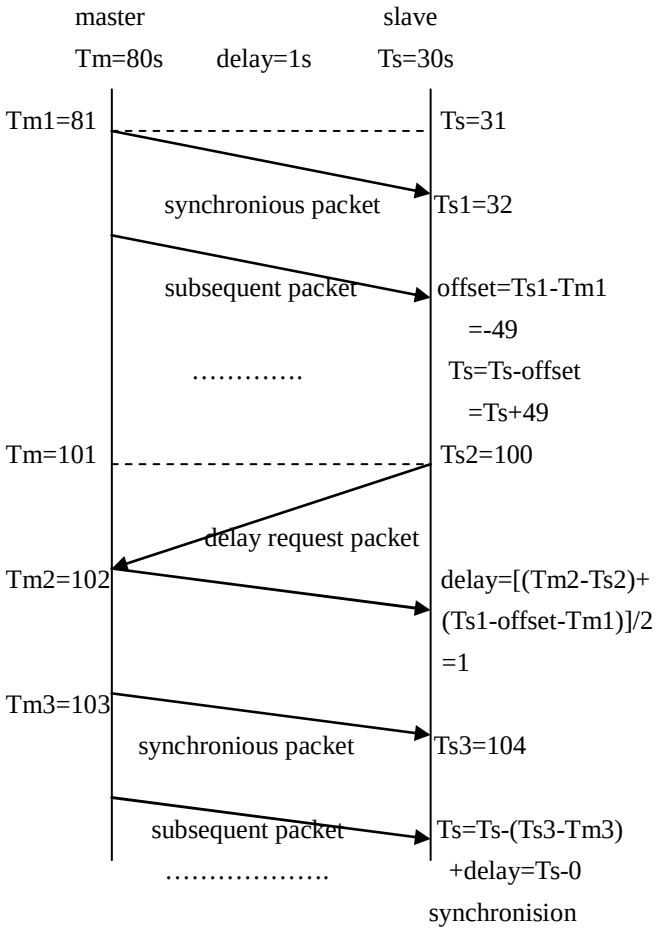

Figure 2. Master-slave synchronization

Offset measurement. Master periodically sends the synchronized packets to all its child nodes by the way of multicast. The accurate expected delivery time of synchronization message is included in the synchronized packets, then, slave record the time when this synchronized packets is received. The time information that is included in the synchronized packets should satisfy the requirements of the lower precision. If the system needs more precise clock synchronization, master should have a specific mechanism, which can detect a packet

transmission time, and then, this specific time will be sent to slave in the form of subsequent packets. By this way, the offset can be calculated between master and slave.

Delay measurement. After adjusting the offset, slave sends a delay request packet to master, master records the time when this delay request packet is received, and then this time information is sent back to this slave, so, slave calculate the transmission delay between the master and itself, and then master-slave clock synchronization is achieved.

3. Results and Analysis

3.1. Test Verify

Fig 3 shows the simplified data flow architecture of the networked control system. Each node connects to the building server room, which is Ethernet switch house. These switches direct data between all the nodes connected to them and to the outside world through an Internet backbone. The control server is directly connected to the physical system, which hosts a PCL-812PG card, a 10/100M adaptive NIC and a Pentium-4 processor running on windows-NT. The control client hosts a 10/100M adaptive NIC and a Pentium-4 processor running on windows-2000.In order to simulate the bandwidth consumption by other equipment that share the same Ethernet networked, three nodes contain 2 kilobyte buffer queue are introduced between the control server and the client. We use VC++ to write experimental programs. Different terminals communicate through calling Socket. Two tested machines separately run client programs and server programs. First, we initialize client and set up connection to the remote server's networked, and then enter the test parameters to begin the test.

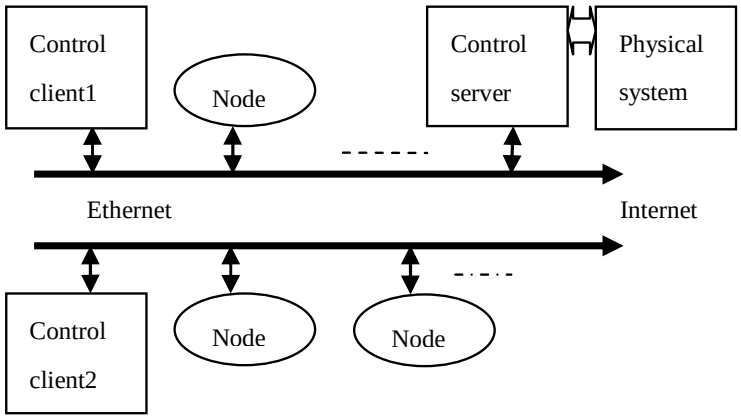

Figure 3. Networked control system architecture

3.2. Test Data

Master clock is the server clock. Slave clock is the client clock, UDP protocol is adopted for all data communication activities between the control server and the client, the datagram is 512byte, the sampling period is 1ms without packet losing, 100data sets will be selected from N data sets at an interval of 10min. Experiment performs for 12 hours. In the network environment of 10M rate, Table 1 shows the test results. In the network environment of 100M rate, Table 2 shows the test results.

Table 1. Pre-test Result

Clock	Master[ms]	slave1[ms]	slave2[ms]
Load	normal load (non-packet retransmission)		
Max-deviation	0	196	187
Min-deviation	0	24	20
Aver-deviation	0	70	66

Table 2. Pre-test Result

Clock	Master[ms]	slave1[ms]	slave2[ms]
Load	normal load (non-packet retransmission)		
Max-deviation	0	934	864
Min-deviation	0	189	180
Aver-deviation	0	374	348

Test result shows that without the use of synchronization algorithms. In the network environment of 10M rate, an average clock deviation from the master to the slaver is about 70milliseconds. In the network environment of 100M rate, an average clock deviation from the master to the slaver is about 374milliseconds. The synchronous precision does not suitable for the requirements of real-time networked control system. Therefore, the least square method is used to solve the problem.

3.3. Part of Program

Part of fitting straight line program

```
FitLine(double x[], double y[],int counts)
{double mx=0;
double my=0;
for(int i=0;i<counts;i++){mx+=x[i];my+=y[i];}
mx=mx/counts;
my=my/counts;
double *xd;
xd = new double[counts];
double *yd;
yd = new double[counts];
double xxsum=0;
double xysum=0;
for(int i=0;i<counts;i++)
{xd = x - mx;yd = y - my;
xxsum = xxsum + xd*xd;
xysum = xysum + xd*yd; }
double k,c;
k = xysum/xxsum;
c = my - k*mx;}
```

Part of synchronization program

```
……..
Whether or not(double offset, double exp)
{int flag;
if(fabs(offset)<=exp) flag=0;else flag=1;
return(flag);}
…….
Synchronization(int flag)
{if(flag)  update_clock();
else wait receivemsg();}
……
```

3.4. Verification

Table 3 shows the test results under the network environment of 10M rate, an average clock deviation from the master to the slave is about 9 microseconds, synchronization accuracy is less than 25 microseconds.

Table 4 shows the test results under the network environment of 100M rate, an average clock deviation from the master to the slave is about 134 microseconds, synchronization accuracy is less than 365 microseconds.

Table 3. Synchronization-test Result

Clock	Master[us]	slave1[us]	slave2[us]
Load	normal load (non-packet retransmission)		
Max-deviation	0	25	21
Min-deviation	0	0	4
Aver-deviation	0	10	8

Table 4. Synchronization-test Result

Clock	Master[us]	slave1[us]	slave2[us]
Load	normal load (non-packet retransmission)		
Max-deviation	0	365	321
Min-deviation	0	0	62
Aver-deviation	0	145	123

PTP Results. Table 5 shows the test results under the network environment of 10M rate, an average clock deviation from the master to the slave is about 7 microseconds, synchronization accuracy is less than 24 microseconds.

Table 5. PTP-test Result

Clock	Master[us]	slave1[us]	slave2[us]
Load	normal load (non-packet retransmission)		
Max-deviation	0	20	24
Min-deviation	0	0	5
Aver-deviation	0	6	8

PTP Results. Table 6 shows the test results under the network environment of 100M rate, an average clock deviation from the master to the slave is about 94 microseconds, synchronization accuracy is less than 304microseconds.

Table 6. PTP-test Result

Clock	Master[us]	slave1[us]	slave2[us]
Load	normal load (non-packet retransmission)		
Max-deviation	0	280	304
Min-deviation	0	0	62
Aver-deviation	0	85	102

4. Conclusion

After the synchronization of master clock, in the network environment of 10M rate, the average phase error of the master clock is about 9 microseconds, in the network environment of 100M rate, the average phase error is about 134 microseconds. These results indicate that a simple least-squares method can satisfy the requirements in real-time networked control system. The method is simple to be designed and easy to be implemented, and that the synchronization clock accuracy reaches sub-microsecond grade, which is the same as the PTP.

Acknowledgements

The authors gratefully acknowledge financial support from Liaoning University of Technology of china under Grant Number X207218.

References

[1] Shames, AN Bishop. Relative clock synchronization in wireless networks. *IEEE Communications Letters*. 2010; 14(4): 348-350.
[2] S Johannessen. Time synchronization in a local area network. *Control Systems Magazine*. 2004; 2: 61-69.
[3] P Blum and L Thiele. *Clock synchronization using packet streams*. in Proc. of 16th International Symposium on Distributed Computing 2002. Toulouse. France. 2002: 1-8.
[4] M Hashimoto, H Hashizume, Y Katoh. *Design of dynamics for synchronization based control of human-robot interaction*. in Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics ROBIO'06. Kunming. China. 2006: 790-795.
[5] MY Ma, L Hu, JD Wu, XN He, H Ma. *Synchronization analysis on converters with distributed control*. in IECON 2006-32nd Annual Conference on IEEE Industrial Electronics. 2006: 2232-2237.
[6] D Mills. Improved Algorithms for Synchronizing Computer Networked Clocks. *IEEE/ACM Trans networkeding*. 1995; 3(3): 245-254.
[7] John Eidson. *Measurement, Control, and Communication Using IEEE 1588*. London. Springer-Vertag New York Inc. 2006.
[8] J Nilsson. *Real-time Control systems with Delays. PH. D. Dissertation*. Sweden & Lund Institute of Technology. Department of Automatic Control; 1998.
[9] L Ljung. *System Identification-Theory for the User*. edtied by Prentice-Hall, Inc. Englewood Cliffs. NJ. 1987.
[10] IEEE Standard Association. 1588-2002. IEEE Standard for a precision clock synchronization protocol for networked measurement and control systems. New York. *IEEE Press*. 2002.
[11] Yan Zhang, Hexu Sun. Application of IEEE 1588 in Real-Time Industrial Ethernet. *Microcomputer Information*. Chinese. 2005; 21(9): 19-21.