A Novel Shared-Clock Hybrid Scheduling Algorithm based on Controller Area Network

Biyun Yan[a], Yehua Wei[a, b]
[a] Institute of physics and information science, Hunan Normal University, Changsha 410081, China
Email: yanbiyun0226@163.com

[b] College of Computer, National University of Defense Technology, Changsha 410073, China
Email: yehuahn@163.com

***Abstract*— The CAN protocol is widely used in the distributed embedded systems. It has been shown that a shared-clock scheduling algorithm can be used along with CAN to implement time triggered architecture. However, the shared-clock scheduling algorithms are mostly used along with the time triggered co-operative schedulers, despite these algorithms provide reliable solutions for many applications, they suffer some key limitations. This paper proposes a shared-clock hybrid scheduling algorithm which employs the time triggered hybrid scheduler. And then the performance of the novel algorithm is analyzed. Finally, a simulation model is built by using TrueTime toolbox in MATLAB platform. The results indicate that the scheduler can improve clock synchronization accuracy, reduce the jitter, and strengthen the real-time character of the system.**

***Index Terms*— Controller Area Network, Shared-Clock, Time-Triggered, Co-operative, Hybrid, Jitter**

I. INTRODUCTION

The CAN protocol has been widely used in automotive and other industrial areas [1]. As a result of its popularity, most modern microcontroller families have members with on-chip hardware support for this protocol. CAN protocol could still be a possible solution due to its deep roots in the automotive electronics industry as well as its simplicity [2], low implementation costs and availability.

In addition, development and achievement gained with CAN over the past years allows the creation of extremely reliable systems using this protocol. A key part of the scheduler is the scheduling algorithm which determines the order in which the tasks will be executed by the processor. More specifically, a scheduling algorithm is the collection of rules that, at every point while the system is running, determines which task must be allocated the resources to execute. Over recent years, time-triggered software architectures have received considerable attention [3]. For multi-processor distributed embedded systems, it has been proved that a "Shared-Clock" (S-C) scheduling algorithm can be used along with CAN protocol to implement time-triggered architectures.

With the research of these years, people have proposed a series of TTC-SC scheduling algorithms. But most of the present shared-clock scheduling algorithms are used along with the TTC schedulers. One shortage of using TTC scheduler is the lack of elasticity. Another issue with TTC scheduler is that the task schedule is usually calculated based on estimates of worst case execution time of the running tasks. If such estimates prove to be error, that may have a serious impact on the whole system behavior [4]. Moreover, one of the primary problems that concern people about the reliability of cooperative scheduling is that long tasks may have a fetal impact on the responsiveness of the whole system [5]; this may be a significant drawback in networks requiring a low tick interval. A feasible solution to these problems is to use a Time-Triggered Hybrid (TTH) scheduler in which a limited degree of pre-emption is allowed.

The rest of paper is organized as follows: In section II, we describe the related works about scheduling algorithm. Section III reviews the schedulers that are referred to in this paper. Section IV introduces the novel TTH-SC scheduler. Scheduler performance is analyzed in section V. Simulation for the purpose of comparison of the novel and present schedulers are presented in section VI. The overall papers conclusions are drawn in section VII.

II. RELATED WORKS

Researchers of embedded systems have proposed various scheduling algorithms that can be used to handle tasks in real-time applications. The selection of appropriate scheduling algorithm for a set of tasks is based upon the capability of the algorithm to satisfy all timing constraints of the tasks: where these constraints are derived from the application requirements. Examples of those common scheduling algorithms are: Rate Monotonic, Deadline Monotonic, Earliest Deadline First, Least Laxity First, and Shared-Clock schedulers.

This work is supported by the National Natural Science Foundation of China (Grant No. 61100215, No.61379115, No. 61311140261), the Project of Hunan Province Science and Technology Program (Grant No.2011GK3182), the Project of Hunan Normal University Young Outstanding Talents Developing Program (Grant No.ET13103).

doi:10.4304/jcp.9.2.373-379

The original S-C scheduling protocols were proposed in 2001 [1], each protocol employs a TTC scheduler and S-C algorithm on top of CAN network. In a more recent study, a collection of possible implementations of the S-C protocol including those presented in were compared and documented [6-7]. Such protocols will be referred to as "TTC-SC1", "TTC-SC2", "TTC-SC3", "TTC-SC4" and "TTC-SC5" protocols, see [7] for more detail. For example, the TTC-SC1, TTC-SC2 and TTC-SC3 schedulers suffer high levels of jitter which may influence the behavior of time-critical systems. In addition, the TTC-SC1 and TTC-SC2 schedulers do not support direct communication between network slave nodes which resulting in comparatively long slave-to-slave message latencies [7]. To reduce jitter and slave-to-slave message latencies, the TTC-SC4 scheduler was proposed; however, such a protocol required a long tick interval and one additional microcontroller. The time-bandwidth utilization in TTC-SC5 scheduler is degraded, due to the scheduling of two messages in each tick interval rather than one message. Nevertheless, in many practical embedded systems, the task duration is extremely short [8]. In most cases, TTC-SC3, TTC-SC4 and TTC-SC5 require all the slaves to reply within one tick interval. As such, the following relationship must hold:

Tick interval > Time take for all ACK messages to be received by the Master

That is the tick interval of the system is related to the number of slaves, and the size of each of the slave's acknowledgement (ACK) messages. That may be the primary drawback of the all proposed schedulers. This paper proposes a novel TTH-SC scheduler which attempts to address the key limitations in the previous protocols while maintains high resource efficiency.

III. Preliminaries

In this section, we will deal with the original S-C algorithm and the time-triggered hybrid scheduler, and then introduce a TTC-SC scheduler in Section C.

A. Shared-Clock (S-C) Algorithm

The S-C scheduling protocol was operated on CAN application layer, as a software platform to achieve time-triggered communications between the nodes connected in the network. It was aimed to provide simple and low-cost software framework architecture for time-triggered systems without requiring other specialized hardware. The S-C scheduler operates as Fig.1. On the Master node, a scheduler operates and the system is driven by periodic interrupts generated from a inner timer. However, on the slave nodes, the slave scheduler is driven by interrupts generated through the arrival of periodic "Tick" messages sent from the Master node rather than on-chip timer. Thus, all nodes of the system will be synchronized according to the master clock [1].

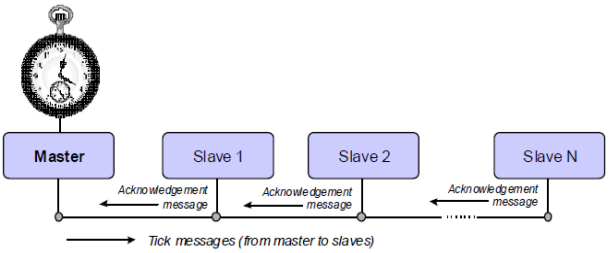

Figure1. Architecture of S-C scheduler

On the whole, the S-C scheduler is extremely simple and allows a number of low cost but effective failure handling mechanisms at the same time. The network communications follow a Time-Division Multiple Access (TDMA) protocol, and the system behavior is highly predictable and reliable. In such a scheduling algorithm, the Tick message holds data for a specific slave or a group of slaves. The first byte of the transmitted data is therefore reserved for the slave or group identifier (ID) to which the tick message is addressed. Only the addressed slave or group slaves have to reply an "ACK" message to the Master after the Tick message is received [1].

B. Time-Triggered Hybrid Scheduler

Over recent years, time-triggered software architectures have received enough attention. A Time Triggered scheduler can be regarded as a very simple operating system which calls tasks periodically during the system is operating. A co-operative scheduler only supported a single-tasking process, if a higher priority task is ready to run while a lower priority task is running at the same time, the former task cannot be broke until the latter one completes its execution period.

While a time triggered hybrid scheduler (TTH) provides a limited, simple but efficient, multi-tasking capability. That is to say, only one task in the whole system is set to be preemptive, this task is also viewed as "highest-priority" task, while other tasks in the system are running co-operatively [4]. A TTH scheduler combines the strengths of the pre-emptive with the co-operative schedulers, and without requiring complex mechanisms for dealing with shared resources at the same time. It is a possible solution to the system that both have long infrequent and short frequent tasks.

C. TTC-SC Scheduling Algorithm

In the TTC-SC algorithm, more than one slave node is allowed to reply the message within one Tick interval. Each moment a Tick message is sent from the Master, an ID is also sent within the Tick message. The TTC-SC scheduler allows that messages sent from the slave nodes can be broadcasted to both Master and all other slave nodes in the network. However, with TTC-SC scheduler, it is possible to have more than one slave reply to each ID. In this situation, we let the CAN controller deal with any message collisions that may appear. The Master node then checks that the appropriate slaves have replied to a designated Tick ID before transmitting the next Tick message. TTC-SC scheduler in this paper is corresponding to the TTC-SCC3 scheduler in reference [7].

For example, assume a four-node system is to be implemented on a single CAN bus. Fig.2 shows the typical message exchange on the CAN network.

IV. Shared-Clock Hybrid Scheduling Algorithm

At present, the shared clock scheduling algorithms are used in conjunction with the TTC scheduler. It is difficult to ensure some significant tasks in real time performance; and should not be applied to the system that requires short tick interval and high clock synchronization accuracy. Thus, we propose a novel shared clock hybrid scheduling algorithm (TTH-SC), which combines the clock scheduling algorithm with the time-triggered hybrid scheduler.

A. Task Schedule in Slave Node

In shared clock hybrid scheduling algorithm, the TTH scheduler implemented in each individual node to achieve time triggered operations of scheduled tasks. Assumed the task that slave node should execute after receiving the tick message (such as an interrupt or send an "ACK" message) has the highest priority, namely the task is pre-emptive, the other tasks are co-operative. When the slave node received the tick message sent from the master node, if the co-operative task being executed, then the processor must respond to tick message and generate an interrupt in order to achieve clock synchronization of the system. In addition, if the ID of slave node matches the ID of tick message. It also needs to send an acknowledgement message to the master node via CAN bus, and then continue to execute the interrupted task. By doing so, we can solve the key limitations of present algorithms. The TTH-SC scheduler can be applied to the systems with the requirement of short tick interval and high clock synchronization accuracy. Fig.3 is an example of scheduling algorithm on the slave node, assumed the task P is preemptive task, C1 and C2 are cooperative tasks, and the execution time of C1 and C2 is much longer than the tick interval.

In the TTH-SC scheduler, we know that when the next pre-emptive task is able to execute, the processor will – sometimes - be in "idle" mode when the timer interrupt generates; or be in "busy" mode, thus the time taken to respond to the pre-emptive task is variable, and hence the response jitter occurs [9]. In this case, we aim to reduce the jitter level by ensuring that the processor is always in idle mode, when the pre-emptive task is due to be executed. We can put the processor into idle mode just before the timer overflow (linked to the dispatch of the pre-emptive task) is triggered, by using an additional on-chip timer to invoke a "Sleep" ISR. Then the response jitter would be degraded. The slave node's scheduling algorithm code shown in the following Table I.

B. Data Transmission between Nodes

Because of CAN protocol employs "Non Return to Zero" (NRZ) coding for bit representation, a drift in the receiver's clock may occur when a long sequence of identical bits has been transmitted on the bus. Such a drift might, in turn, result in message corruption and error. To avoid the possibility of such a disaster, the CAN communication protocol uses a bit-stuffing mechanism which operates as follows. After five consecutive identical bits have been transmitted in a given frame, the sending node adds an additional bit of the opposite polarity afterwards. All receiving nodes remove the inserted bits to recover the original data. Whilst providing an effective mechanism for clock synchronization in the CAN hardware, the bit-stuffing mechanism causes the frame length to become a complex function of the data contents. When using (for example) 8-byte data and standard CAN identifiers, the minimum message length will be 111 bits (without bit stuffing) and the maximum message length will be 135 bits (with the worst-case level of bit-stuffing). At the maximum CAN baud rate (1Mbit/sec), this translates to a possible variation in message lengths of 24 us [10].

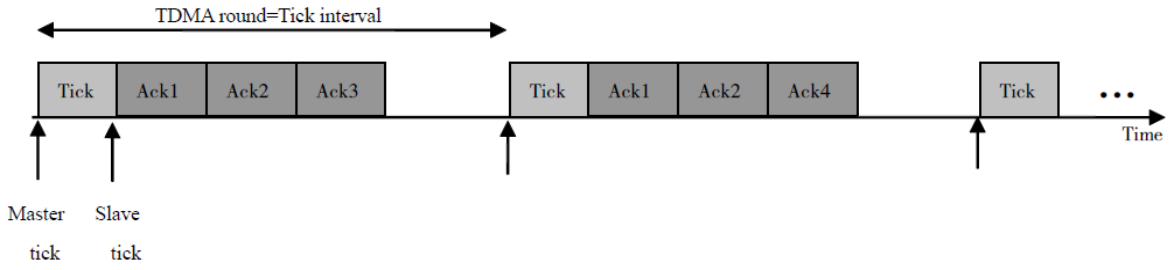

Figure2. Architecture of TTC-SC scheduling algorithm

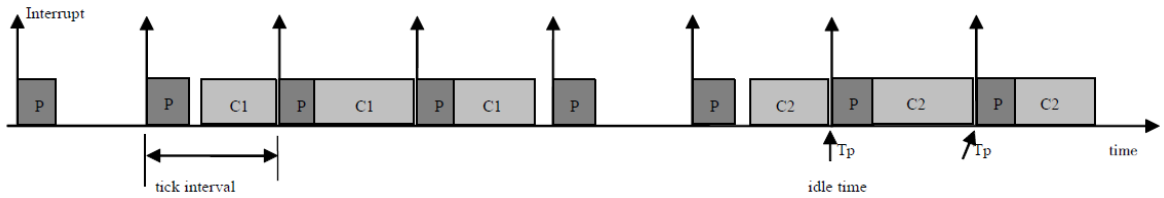

Figure3. The TTH scheduler in slave node

TABLE I.
Shared-Clock Hybrid Scheduling Algorithm

```
while(1)
      {
        C_Task( ) ;// execute cooperative task
        Sleep( );// put the processor into idle mode
      }
void Slave_P_Task( void)
    {
       #define Slave_Group_ID ox02 //define group ID
      #define Network_no_error (1)
      Network_error_pin = Network_no_error ;
                    // set bit after received the tick message
      Interrupt_Clock_Update( ) ;    // clock synchronization
      if (Tick_Group_ID == Slave_Group_ID )
                     // whether the ID matches or not
        {
            SCH_Send_Ack_Message_To_Master( );
                    // send acknowledgement message
        }
    }
void P_Dispatch_ISR (void)
    {
       Sleep( );   // put the processor into idle mode
      Slave_P_Task( ) ; // execute preemptive task
    }
```

To explain the novel TTH-SC scheduling algorithm better, assume a five-node system as illustrated in Fig.4. The figure shows how slave ACK messages can be scheduled in a simple TTH-SC scheduling algorithm, where the two slaves are allowed to transmit in the same tick interval. In this situation, the TDMA round is equal to the two tick interval, ACK N represent the acknowledgement message of slave N, Tick represent the tick message.

When using TTH-SC scheduler, the Master node is configured to send the Tick messages without containing any data. These messages synchronize the network and only function as the synchronization messages. Thus, the Master node simply generates the "heartbeat" of the whole network, but does no data transmitting process with slave nodes at the same time. This algorithm allows the Tick message to have a comparatively reduced and fixed data content, which results in a constant CAN Tick message length. Thus, transmission jitter caused by the variable Tick messages length can be largely reduced. In this case, the Master will only use one data byte for "Group ID" to which particular messages are sent, and the other seven data byte would be empty.

V. THE PERFORMANCE OF TTH-SC SCHEDULER

In this section, we discuss the performance of the TTH-SC scheduler. At first, we evaluate the node failure detection time in section V-A. Transmission time and jitter are discussed in section V-B.

A. Failure Detection Time

In the TTH-SC scheduler, the Master node has the responsibility to check the status of all slave nodes and deal with any node-failure. Since the TTH-SC scheduler is partly adapted from the TTC-SC scheduler, Failure detection time is certainly reduced here. Fig.5 illustrates an example where slave 1 suffers a failure as soon as it has sent its ACK message.

The worst-case time that the Master node takes to detect a failure on the slave node is calculated as follows:
Worst-case failure detection time

$$= \text{TDMA} + \text{Tick} - M = (N+1)\,T - M \qquad (1)$$

Where M is the Master Tick message length, T is the tick interval. As the tick interval is shorter than TTC-SC3 and TTC-SC4 scheduler, the worst case time is reduced.

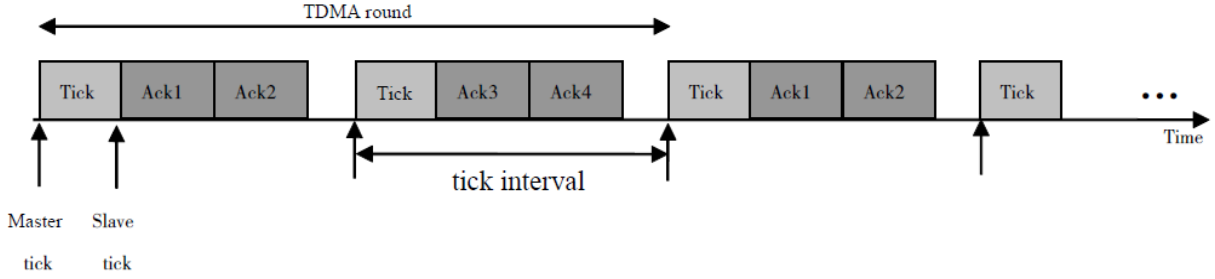

Figure4. Architecture of TTH-SC scheduling algorithm

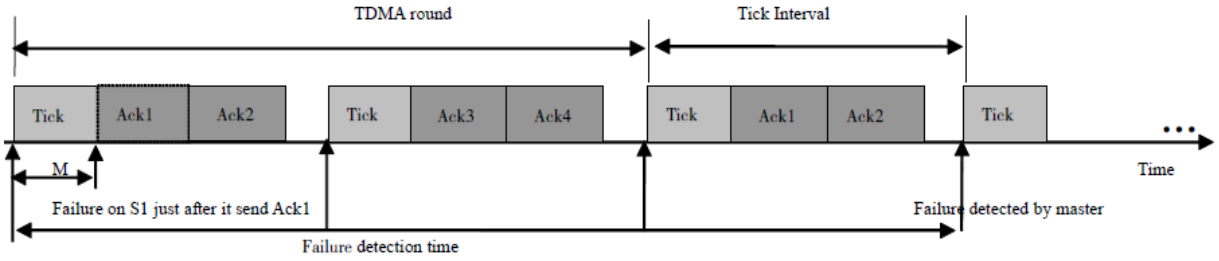

Figure5. Failure detection time of TTH-SC

B. Transmission Time and Jitter

Jitter is a term which describes variations in the timing of activities [11]. In any S-C scheduler, Tick messages are sent from the Master at each tick interval to drive the Slave schedulers. If such messages have variable lengths, this is likely to generate jitter in the process of tasks running in the slave nodes. The TTH-SC scheduler has more advantages while maintaining the strengths of the proposed TTC-SC scheduler. Jitter caused by CAN bit-stuffing is significantly reduced. Tick message transmission time is also reduced due to the message has empty data. Moreover, messages sent by a given slave will be broadcasted to all other slaves in the network, allowing a direct communication between any communicating slaves. So the slave-to-slave message latency is also comparatively reduced.

VI. SIMULATION

The TrueTime toolbox in the MATLAB environment provided a very good platform for the research of network control system, the dynamic process of multi-processor distributed real-time control system, and control task execution. In addition, simulation environment of network interaction can be built by TrueTime toolbox [12]. At the same time, it does not need actual network connection, and its advantages are relatively simple, low cost and code open. Meanwhile, owing to it based on Simulink environment of the Matlab, it is very easy to design the fictitious network control system. The TrueTime is a very good platform for delay compensation algorithm of network control system and scheduling algorithms of different resources. There are four main modules in the TrueTime toolbox, namely the TrueTime Kernel module, TrueTime Network module, TrueTime Wireless Network Module and TrueTime Battery module. Connecting the modules in TrueTime and the common modules in Simulink, the corresponding real-time control system or network control system can be built [13].

A. Simulation Model

The simulation model of the network control system built by using TrueTime toolbox is shown in the Fig.6. The system includes five network nodes, and each of them is represented by TrueTime kernel module. The system contains a master node (Node 2), which is used to produce the “heartbeat” of the network, send the tick messages to all slave nodes, the master node is set to clock driven mode, and the other nodes are set to event-driven mode. Node 5 is periodic sampling from the controlled object, and through the network module, the samples will be sent to node 4. The task of node 4 is calculating the control signals and sending the results to node 3, and the node 3 executes the control signals later. The network communication model used in this paper is CSMA / AMP (CAN), CAN baud rate is set to 800Kbps and the tick interval is 1ms. We try to compare the behavior of the TTC-SC scheduler with the TTH-SC scheduler by running the simulation model.

B. Simulation Results

The results obtained from this simulation are showed in the Fig.7. From the figure we know that when comparing with the TTC-SC scheduler, the tick message transmission time and jitter of TTH-SC scheduler have largely declined. Thus, the accuracy of clock synchronization of the whole network system will improved greatly.

Fig.8 is the comparison of the reference signal with the measurement signal of node 4. It can be seen from the figure that when TTH-SC scheduler was used in the system, the measurement signal is approximately the same as reference signal (square wave), the task deadlines are met, and the system performance is guaranteed. The results indicate that the TTH-SC scheduler can be applied to the system that has both short frequent and long infrequent tasks.

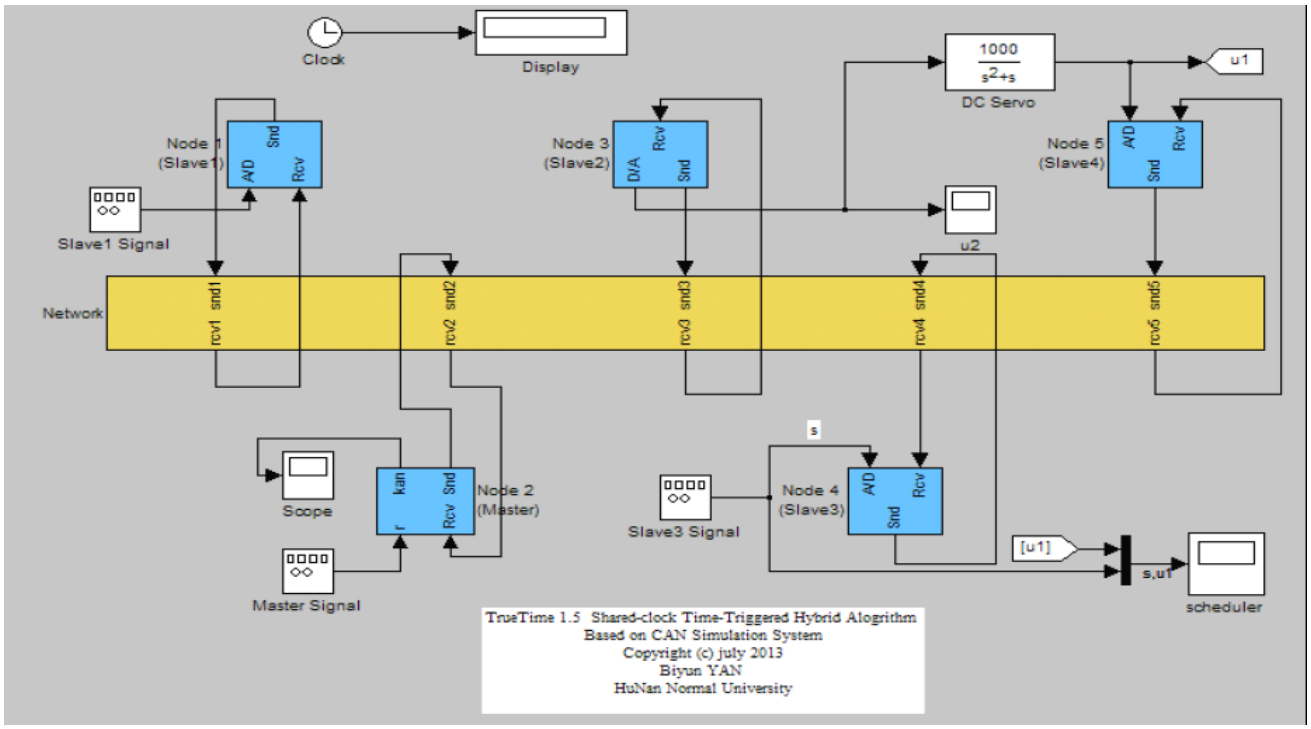

Figure6. Simulation Model

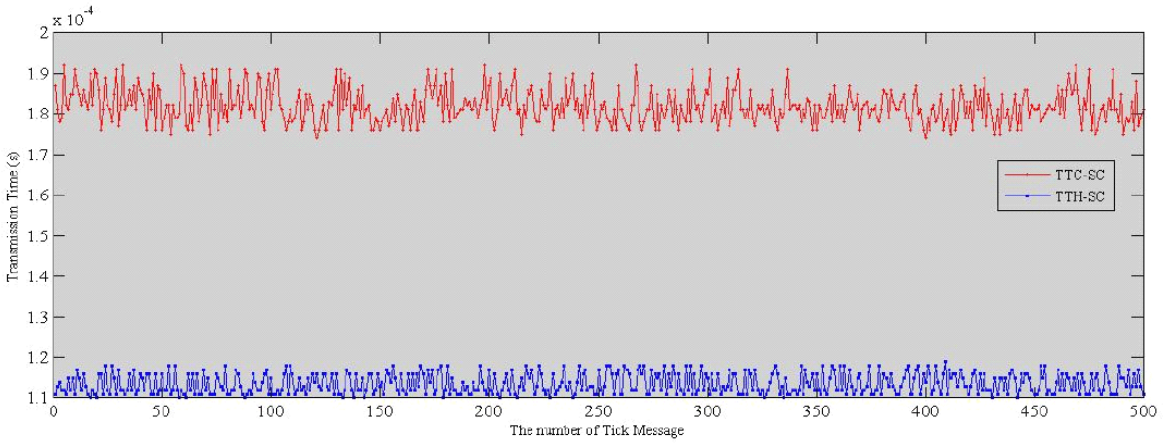

Figure7. Transmission Time

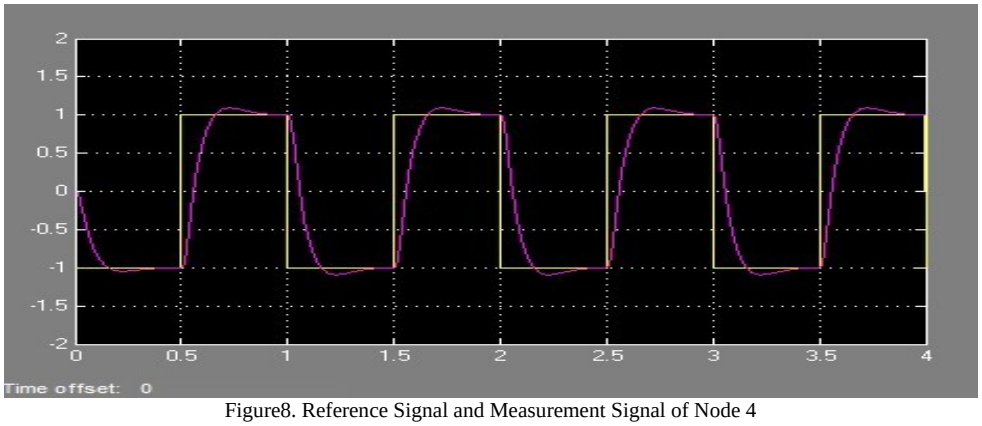

Figure8. Reference Signal and Measurement Signal of Node 4

VII. CONCLUSIONS

The work presented in this paper began by reviewing a series of developed S-C scheduling algorithms and some related works. Despite that such protocols provide reliable solutions for many applications, they also suffer some key limitations. Thus, we propose a novel TTH-SC scheduler, which provides a limited multi-tasking

capacities and a reduced jitter level in the message transmission, while maintaining low slave-to-slave message latencies and high resource efficiency.

Then we build the simulation model of network control system by using TrueTime toolbox, the simulation results show that the algorithm can reduce jitter, improve the accuracy of clock synchronization. The TTH-SC scheduler would be an attractive choice for the systems which require high clock synchronization accuracy and a short tick interval, and also be a reliable solution to the system which has both short frequent and long infrequent tasks.

ACKNOWLEDGMENT

The authors are grateful to the anonymous referees for their valuable comments and suggestions to improve the presentation of this paper.

REFERENCES

[1] Pont M J, "Patterns for Time-triggered Embedded Systems: Building Reliable Applications with the 8051 Family of Microcontrollers," Addison-Wesley Professional edition, July 12 2001, pp. 539-522.

[2] Gerardine Immaculate Mary, 1 Z. C. Alex, et al, "Response Time Analysis of Messages in Controller Area Network: A Review," Journal of Computer Networks and Communications, Volume 2013, http://dx.doi.org/10.1155/2013/148015.

[3] Xia Jiqiang, Zhang Chuansen, Bai Ronggang,"Real-time and reliability analysis of time triggered CAN-bus," Chinese Journal of Aeronautics, 2013, 26(1), pp. 171–178.

[4] Mouaaz Nahas,Ahmed M. Nahas, "Ways for implementing highly-predictable embedded systems using time-triggered co-operative(TTC) architectures," 2012, www.intechopen.com.

[5] Pont, M.J, "Supporting the development of time-triggered cooperatively scheduled (TTCS) embedded software using design patterns," Informatica, 2003, 27, pp. 81-88.

[6] Devaraj Ayavoo, Michael J. Pont, Michael Short, et al. "Two novel shared-clock scheduling algorithms for use with CAN-based distributed systems,", Microprocessors and Microsystems, 2007, 31(5), pp. 326-334.

[7] Mouaaz Nahas. "Developing a Novel Shared-Clock Scheduling Protocol for Highly Predictable Distributed Real-Time Embedded Systems," American Journal of Intelligent Systems, 2012, 2(5), pp. 118-128.

[8] Amir Muhammad, Michael J. Pont, "Comments on: Two novel shared-clock scheduling algorithms for use with 'Controller Area Network' and related protocols," Microprocessors and Microsystems, 2011(35), pp. 81–82.

[9] A. Maaita, M.J. Pont, "Using "planned pre-emption" to reduce levels of task jitter in a time-triggered hybrid scheduler," in Proceedings of second UK embedded forum, 2005, pp. 18-35.

[10] Mouaaz Nahas, Michael Short, Michael J. Pont, "The impact of bit stuffing on the real-time performance of a distributed control system," CAN in Automation, 2005, pp. 101-107.

[11] M. Nahas, M.J. Pont, A. Jain, "Reducing task jitter in shared-clock embedded systems using CAN," in Proceedings of first UK embedded forum, 2004, pp. 184-195.

[12] Dan Henriksson, Anton Cervin, Karl-Erik Årzén, "TRUETIME: Real-time Control System Simulation with MATLAB/Simulink".

[13] Daogang Peng, Hao Zhang, Conghua Huang, et.al. "Study of Immune PID Networked Control System Based on TrueTime," JOURNAL OF NETWORKS, VOL. 6, NO. 6, JUNE 2011, pp. 912-915.

Biyun Yan corresponding author, received his B.S. degree June 2012, is currently a postgraduate student and working towards his M.S. degree at Hunan Normal University, China. His current research interest includes network communication and embedded system.

Yehua Wei, born in 1979, received his Ph.D. degree from the school of computer and communication of Hunan University, Changsha, China in 2009. He is an associate professor at the School of physics and information Science, Hunan Normal University, China. His main research interests include wireless networks and embedded computing.