TTE时间同步协议关键算法研究和仿真分析

黄 韬，陈长胜

(中航工业西安航空计算技术研究所 西安 710119)

【摘要】针对一种用于安全关键领域的实时网络协议——时间触发以太网(time triggered ethernet，TTE)，分析研究TTE的时钟同步协议，对时钟同步流程以及协议涉及到时序保持算法、集中算法、时钟同步算法进行研究和分析。建立网络交换机和节点机的模型，构建时间触发网络通信仿真平台，对时间触发以太网时钟同步协议进行仿真分析，验证了该时间同步算法能完成时间同步，并且精度保证在95 ns以内。

关 键 词 算法; 时钟同步; 仿真; 时间触发
中图分类号 TP393.1 文献标志码 A doi:10.3969/j.issn.1001-0548.2014.03.021

TTE Synchronization Protocol Key Algorithm and Simulation Analysis

HUANG Tao and CHEN Chang-sheng

(Aeronautics Computing Technique Research Institute Xi'an 710119)

Abstract The paper focus on a real-time network, time-triggered Ethernet (TTE), which is used in safety-critical systems. The clock synchronization protocol related to the process of clock synchronization is researched and the permanence algorithm, compression algorithm, and clock synchronization service are analyzed. Then, the simulation platform including the switch model and node model is build. Finally the effectiveness and efficiency of time-triggered Ethernet clock synchronization are simulated and analyzed.

Key words algorithm; clock synchronization; simulation; time-triggered

以太网技术发展30多年来，其带宽大、成本便宜、结构简单成为大家的共识，其应用已渗透到了社会的各行各业：生产控制、船舶、机载、车载、办公场所、安全关键系统等。但以太网应用于高实时性、安全关键系统时无法解决时间确定性问题，对此以太网应用于这些场合必须解决该键问题。TTE技术的提出解决了的时间确定性问题[0]，同时支持多种(单或多通道)通信方案供用户选择[1][0]，能够满足高实时要求[0]、安全关键系统[0]的需求，是极具发展前景的实时网络通信技术。

“时间触发”是指网络具有统一的时间基，所有的通信都是按照全局时间进行调度。网络上的所有活动都是随着时间的前进而有计划地调度。TTE中的消息主要面向状态，而不是事件，每条消息一直保持到状态改变，而状态也只能在规定的时间间隔内改变[0]。所以TTE必须建立一个全局统一的时间，并以此为基础按照预先定义的通信配置表进行时分复用方式的通信。

作为TTE技术的权威，TTTech公司对时间触发以太网给出如下定义[0]：

TTE=以太网+时钟同步+时间触发通信+速率限制传输+保证传输。

TTE传输3种不同优先级的数据帧：时间触发数据、速率限制数据和尽力而为数据，3种数据优先级依次递降。

本文首先分析了已有的时钟同步机制，然后介绍了TTE时钟同步协议过程并深入研究了同步协议的3个关键算法。在仿真平台上通过仿真模型对时间触发以太网的时钟同步协议进行分析和验证。

1 TTE时钟同步协议研究

1.1 时钟同步算法概述

在高性能实时以太网研究内容中，最重要的是全局时钟同步问题。而现有的时间同步算法(如PTP[8-9]、DTP、NTP等)无法满足需求，TTE时钟同步算法[0]的研究显得尤为重要。只有解决了时间同步

收稿日期：2013－05－23； 修回日期：2013－11－20
作者简介：黄韬（1970－），男，研究员，主要从事机载网络技术方面的研究.

问题，才能保证整个系统在TDMA机制下可靠的传输时间触发消息和事件触发消息，保证在一定精度内进行高性能实时以太网数据的可靠传输。所以本地时钟同步的正确性是时间触发网络的基本前提和必须条件。

1.2 工作原理和流程

TTE网络中包含3种角色：同步控制器(synchronization master, SM)，集中控制器(compression master, CM)，同客户端(synchronization client, SC)，它们之间通过传递协议控制帧(protocol control fram, PCF)进行同步。

TTE时间同步流程[0]分为两步，如图1所示。

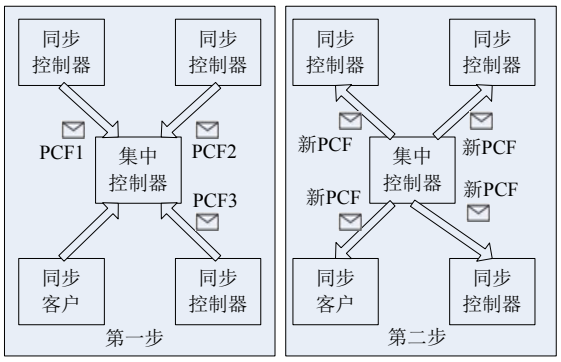

图1 TTE时钟同步流程

同步流程为：

1) SM_i(i表示节点i)向CM发送PCF_i。CM收到各个链路上的不同的PCF(PCF1-PCF3)后，对这些PCF进行时序保持算法，以恢复PCF的接收顺序与其发送顺序一致。然后再对这些PCF所包含的时间信息进行集中处理，取得折中的时间。

2) 将第一步的计算结果写入一个新的PCF中广播给各个节点。网络中的各个节点收到由CM发回的PCF之后，根据PCF帧所包含的时间信息修正自己的本地时间，实现全局的时间同步。

1.3 时钟同步算法详细步骤

结合时间同步构件[0]和全局时钟同步算法[0]，图2为时间同步在各类型节点上的流程详图，基本算法步骤简述如下：

1) 令sm_dispatch_pit=0，当SM_i的sm_clock_i=0, SM就开始发送PCF_i；

2) 在时刻receive_pit_i(记r_i)，CM接收到该PCF_i，并启动消息时序保持算法。在读取r_i后计算cm_permanence_pit_i(即还原帧顺序后的接收时间，记作p_i)，对网络传输延迟造成的影响进行修正；

3) 在CM上的时刻p_i，启动集中算法，计算出cm_compressed_pit(记ct)值和member ship_new的和值；

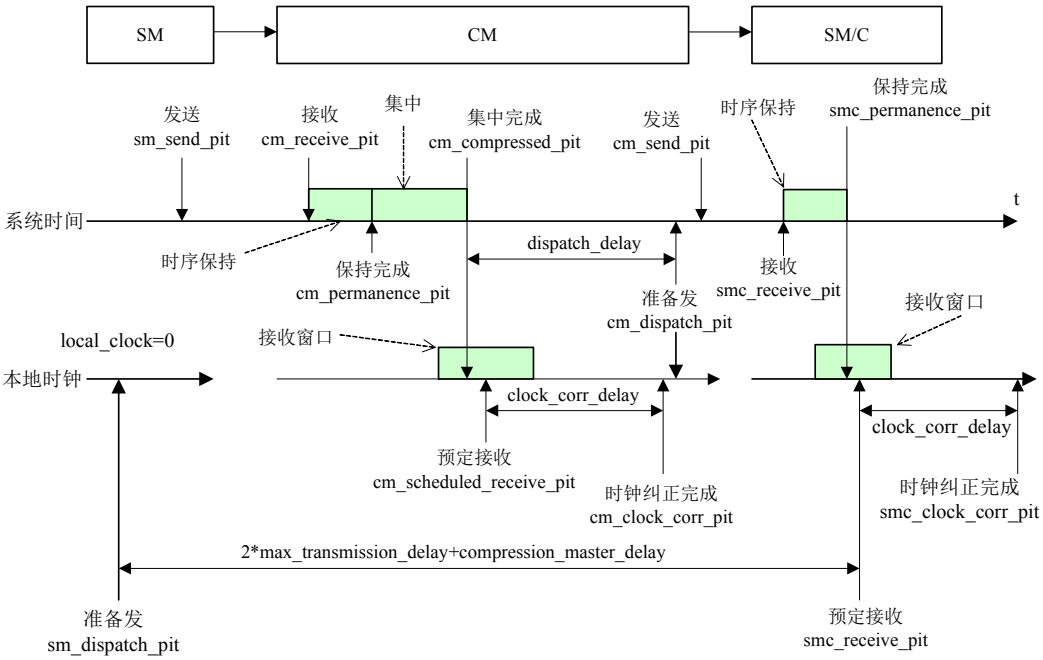

图2 同步算法流程图

4) 根据ct值，经过一个延迟时间clock_corr_delay，CM时钟值到达ct+clock_corr_delay时，进行CM时钟修正并在等待dispatch_delay后产生一个NEW_PCF广播到SM和SC；

5) 在SM/SC上的时刻smc_receive_pit(记r^{cm})，接收到NEW_PCF，并通过时序保持算法得到时序保持时间smc_permanence_pit(记为p^{cm})；

6) 本地时钟到达p^{cm}时，再经过延迟clock_corr_

delay后，在时刻p^{cm}+clock_corr_delay完成时钟修正。

2 关键参数定义

为了完成时间同步算法，有几个重要参数分析如下。

2.1 透明时钟

参照IEEE1588，透明时钟用于记录帧经过网络传输的延时。通过测量帧在各节点的驻留时间以及传输时间，并累加到消息的透明时钟字段。TTE中透明时钟的值保存在PCF的Transparent Clock字段，记录该PCF从源节点到目的节点所经过的传输总延时，记作t_i。

2.2 最大传输延时

最大传输延时表示整个网络系统中任意两个节点之间的最大延时，是一个离线得到的通信网络统计定值，拓扑建好后它是一个定值，计算如下(记作MAX)：

$$\text{MAX}=\max(\text{pcf\_transparent\_clock}_n) \tag{1}$$

2.3 集中算法延时

集中算法延时是集中算法执行的时间开销。为了保证完成时间同步的精度，它不可忽略，分别记作D_{cm}和D_{smc}，具体计算如下：

$$D_{cm}=\text{max\_observation\_window+calculation\_overhead} \tag{2}$$

$$D_{smc}=\text{max\_observation\_window+calculation\_overhead+dispatch\_delay} \tag{3}$$

式中，max_observation_window(最大观察窗口)由系统的容错余度即窗口个数和同步精度(观察窗口大小)决定；calculation_overhead为集中算法的计算处理时间；dispatch_delay为帧的发送延时。

2.4 预定接收时间

预定接收时间PCF_i能够与实际的permanence_pit_i相减便得出时间偏差。该时钟同步采用了时序保持算法，所有帧的传输延迟都是一个定值，并且CM的处理延时也为定值。所以能够预先推出CM的预定接收时间值Scm和Ssmc的预定接收时间值，记作S_{cm}和S_{smc}：

$$S_{cm}=\text{sm\_dispatch\_pit+max\_transmission\_ delay+compression\_master\_delay} \tag{4}$$

$$S_{smc}=\text{sm\_dispatch\_pit+2*max\_transmission\_delay+smc\_compression\_master\_delay} \tag{5}$$

3 时间同步协议算法

3.1 同步算法理论

信道最优判断：若存在多个满足进行时钟同步的帧，则选择membership_new最大的通道中p_i最大值作为有效PCF。公式如下：

$$\text{best\_channel}=\max_{\text{smc\_permanence\_pit}}\left(\max_{\text{pcf\_membership\_new}}(\text{PCF})\right) \tag{6}$$

PCF传输中，PCF_i从SM→CM有传输延时，但是定义整个网络的最大传输延时MAX，以还原PCF的原始发送顺序，时序保持算法定义为：

$$P_i=\text{MAX}-t_i+r_i \tag{7}$$

集中算法是根据 PCF 在接收窗口内的所有 SM 节点时钟偏差计算一个均衡值，集中算法定义为：

$$c=\begin{cases}0 & N=1\\ \dfrac{p_2-p_1}{2} & N=2\\ p_2-p_1 & N=3\\ \dfrac{((p_2-p_1)+(p_3-p_1))}{2} & N=4\\ p_3-p_1 & N=5\\ \dfrac{p_{\max\_k^{th}}-p_{\min\_k^{th}}}{2} & N>6\end{cases} \tag{8}$$

式中，N为接收窗口内的PCF个数；c表示compression_pit；$p_{\max_k}$和$p_{\min_k}$分别表示第k个p值最大的帧和最小的帧。

$$\text{ct}=p_1+c+M \tag{9}$$

设常数M=max-observation_window+ calculation_overhead

由式(8)可见，c就是全网络所有SM节点时钟偏差的一个均衡值，CM根据c计算出ct并取最优ct_{best}相关同步信息整理为一个NEW_PCF。NEW_PCF可说是所有SM发送来的同步信息的一个PCF帧，被CM认为是一个标准的时钟发启的PCF。若CM时间精准，则S_{cm}便是接收该NEW_PCF的时间，而实际接收时间为ct_{best}。故ct_{best}与S_{cm}作差得到时钟修正值clock_corr_{cm}，可以认为是CM被所有SM的信息时钟同步了，计算公式：

$$\text{clock\_corr}_{cm}=S_{cm}-\max\{\text{ct}_{\text{best\_channel}}\} \tag{10}$$

最优信道依据式(6)判断，带入式(7)～式(9)，化简得：

$$\text{clock\_corr}_{cm}=\max\{(t_1-c-r_1)_{\text{best\_channel}}\} \tag{11}$$

由式(11)可知，CM进行时钟同步仅与参数r_1、p_1、c有关，(p_1-c)是接收窗口内PCF的均衡值，$[r_1-(p_1-c)]$是CM本地时钟与标准时钟之间的偏差。

SM/SC时钟修正算法：SM/SC接收了NEW_PCF后，该帧被认为是一个时钟精度的节点发送的，接收时间p_{cm}与预计接收时间S_{smc}作差，得到：

$$\text{clock\_corr}_{smc}=S_{smc}-\text{average}(p^{cm}_{best\_channel}) \quad (12)$$

最优信道依据式(6)判断，式(5)和式(7)带入化简，得：

$$\text{local\_corr}=\text{average}_i(t^{cm}-r^{cm})_{best\_channel}+\text{MAX}+D_{smc} \quad (13)$$

由式(13)看出，SMC的修正值就是NEW_PCF的传输延迟与收到帧的本地时钟值之间差。CM/SM/SC的时钟修正为：

$$\text{local\_clock}=\text{local\_clock}+\text{clock\_corr} \quad (14)$$

3.2 时序保持算法

时序保持算法用于屏蔽网络传输延时对时间同步的影响。引入网络传输延时的因素包括：发送延时、链路延时和接收延时，对于高精度的时间同步，这些延时是必须要考虑的。时序保持算法通过PCF帧中的透明时钟对PCF帧到达时间进行修正，以恢复各个PCF帧的时序关系。

消息时序保持算法的流程如下：

1) 节点收到PCF_i，读取当前的本地时间r_i，同时执行消息时序保持算法；

2) 读取该PCF_i的Transparent Clock字段得到透明时钟值t_i；

3) 运行式(7)完成时序保持算法：

如同上面算法，在接收端通过时序保持算法：首先根据网络拓扑和网络节点发送、接收的处理时间，得到整个网络的最大传输延时MAX，然后按照式(7)计算出时序保持时间p。

3.3 集中算法

由于晶振本身的稳定性以及环境等影响，各个SM时间是不一致的，其PCF帧发送时间也是不一致的。集中算法是CM根据接收到的属于同一个整合周期的所有PCF帧的p值，取一个均衡值ct，并且生成一个NEW_PCF。其memberiship_new中包含所有这些PCF的memberiship_new值。这个PCF随后被广播到各个SM和SC，实现全网的时间同步。

CM接收到PCF后，会根据帧的membership_new域值判断是否进行集中算法(i为membership_new字段的第i位，用于标记SM_i)：$\sum_{i=1}^{32}\text{membershi_new}_i=1$且无PCF开启算法，则进行集中算法；$\sum_{i=1}^{32}\text{membershi\_new}_i>1$，则不进行集中算法。

集中算法的流程如下：

1) 开启一个观察窗口(observation window)，标记为 OW=1；

2) 判断该观察窗口内是否有完成时序保持的 PCF 帧。如果有并且 OW⇐f+1，f 为系统容错的限值，则新开启一个窗口，继续观察；否则进入下一步转到步骤 3)；

3) 收集观察窗口内的所有 PCF 帧：

① 应用式(8)计算得 compression_correction;

② 应用式(9)计算得 compression_pit;

4) 整合这些 PCF 帧的 membership_new 为一个新值，包含所有 PCF 对应位。

3.4 时钟修正算法

CM根据SM发送的PCF计算出整个系统的SM时间偏差均衡值，完成CM的本地时间修正。CM通过集中算法求出集中算法结果ct_{best}与预计时间S_{cm}的差值就是本地时钟与全局时钟的偏差local_corr，根据式(11)和式(14)完成时钟修正。

SM/SC根据CM时间修正后返回的NEW_PCF完成时间修正。SM/SC的S_{smc}提前预知，所以实际接收PCF帧的时序保持时间(此处为最优信道多个最优PCF的均值)与预计时间的差值就是本地时钟与全局时钟的偏差local_corr，根据式(13)和式(14)完成时钟修正。

4 OPNET仿真实现

4.1 仿真场景设计

仿真拓扑结构如图3所示，拓扑结构为双通道星型，节点包括：2个CM交换机节点，4个SC交换机节点，16个SM终端节点，8个SC终端节点；链路采用100BaseT；仿真时间为3 s。

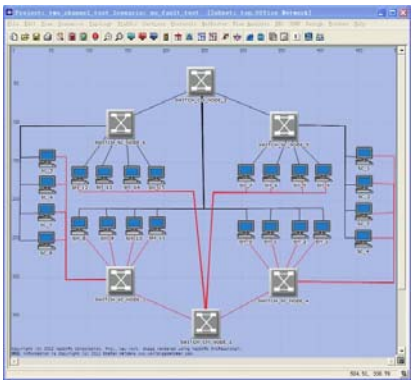

图3 仿真拓扑结构图

场景中的所有节点的local_clock_durat ion取值为(1ns+随机偏差值)，设置网络中所有的节点同时开始启动，也即每个节点的本地时钟都是从零开始计

时，并且始终保持同步状态。进行仿真实验是为了验证执行同步操作之后的每个节点的时钟是否满足设定的精度要求。

4.2　仿真结果分析

在OPNET上本文实现了时钟算法的同步，测试结果显示每个节点都是可以正常同步，时钟修正的值也是正确的。满足预期结果的要求。

图4给出了仿真的速度和所用时间的OPNET截图。由于TTE网络的精确度是纳秒，故每个节点的事件总量相当大，该场景仿真时间3 s内处理的事件数为9.000 8×10^{10}个，平均速度为157 433 events/s，共用时间158 h 48 min。

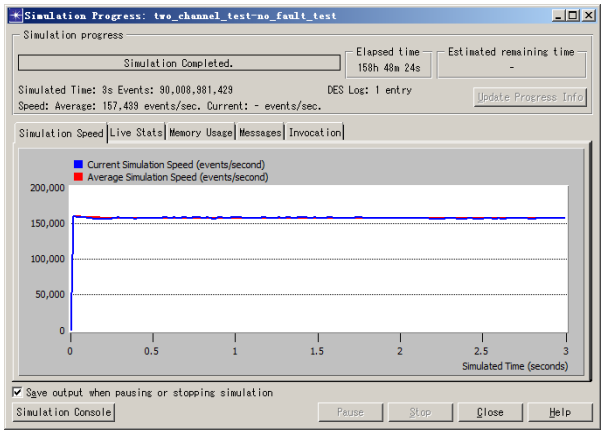

图4　场景仿真速度

图5给出了双通道网络中3种数据吞吐量，由图可知TT数据在发送时的吞吐量为6×10^{6}bps、RC数据在发生时的吞吐量为6.4×10^{6}bps、BE数据在发送时的吞吐量为(4.4×10^7–4.56×10^7)bps，TT:RC:BE吞吐量比大致为1:1:8。

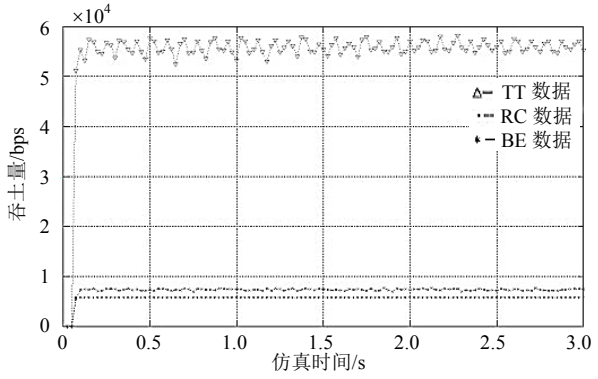

图5　吞吐量统计结果

图6给出了双通道网络结构中TT、RC和BE共3种数据的端到端网络延迟对比情况，由图可知，TT数据的端到端延迟最小；RC数据的端到端延迟较大，其值始终小于2×10^{-5} s；对BE数据的端到端延迟范围统计如表1所示。

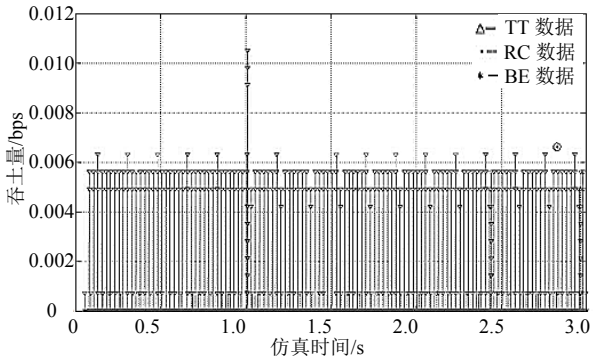

图6　延时统计

表1　BE数据端到端网络延迟

时间区间/s	个数	百分比/(%)
[0, 2×10^{-5}]	17 422	90.12
[2×10^{-5}, 1×10^{-3}]	678	3.51
[1×10^{-3}, 7×10^{-3}]	1 232	6.37

图7给出了双通道时钟精度，由图可知，时钟精度控制在95 ns以内。

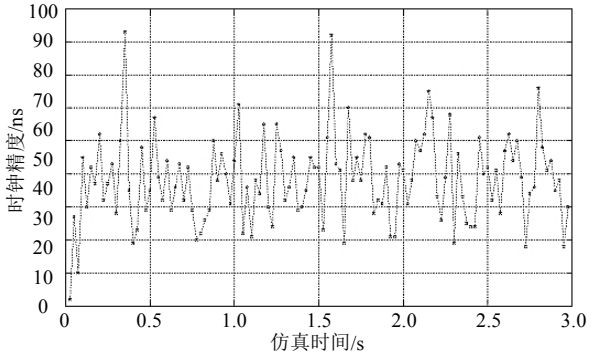

图7　双通道中的时钟精度

通过仿真实现证明，该时间同步算法能完成时间同步，并且精度保证在95 ns以内。时间同步机制能保证整个系统在TDMA的机制下可靠的传输时间触发消息和时间触发消息，能保证在一定精度内进行高性能实时以太网数据可靠地传输。

5　总　结

本文对时间触发以太网中的时钟同步协议进行深入研究和分析，并利用OPNET仿真工具，对TTE时钟同步协议进行仿真分析，对3个关键算法和同步性能进行分析验证，为时间触发以太网的开发应用提供有力的理论支撑。

参　考　文　献

[1] 杨仕平, 桑楠, 熊光泽. 基于Ethernet技术的安全关键实时网络[J]. 软件学报, 2005, 16(1): 122-123.
YANG Shi-ping, SANG Nan, XIONG Guang-ze. Safety critical real-time networks based on ethernet technology[J].

Journal of Software, 2005, 16(1): 122-123.
[2] KOPETZ H. Fault containment and error detection in the time-triggered architecture[C]//Autonomous Decentralized Systems, 2003. [S.l.]: IEEE, 2003: 139-146.
[3] MURDOCK J K, KOENIG J R. Open systems avionics network to replace MIL-STD-1553[C]//Digital Avionics Systems Conference. [S.l.]: IEEE, 2000.
[4] NICOLAS N, SONG Y Q, FRANÇOISE S. Worst-case deadline failure probability in real-time applications distributed over controller area network[J]. Journal of systems Architecture, 2000, 46(7): 607-617.
[5] GÜNTER H, THURNER T. Time-triggered architecture for safety-related distributed real-time systems in transportation systems[C]//Fault-Tolerant Computing, 1998. Digest of Papers. [S.l.]: IEEE, 1998: 402-407.
[6] HOANG, H. Rea-time communication for industrial embedded systems using switched ethemet[C]//Parallel and Dis-tributedProcessing Sysposium．[S.l.]: IEEE, 2004．
[7] HERMANN K, GUNTER G. TTP-A protocol for fault-tolerant real-time systems[J]. IEEE Computer, 1994, 27(1): 14-23
[8] LEE K, EIDSON J C, WEIBEL H, et al. IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems[C]//Sensors for Industry Conference, 2002. [S.l.]: IEEE, 2002: 98-105.
[9] YU Peng-fei, YU Qiang, DENG Hui, et al. The research of precision time protocol IEEE1588[C]//The International Conference on Electrical Engineering. [S.l.]: [s.n.], 2009.
[10] KOPETZ H, ADEMAJ A, GRILLINGER P, et al. The time-triggered Ethernet (TTE) design[C]//Object-Oriented Real-Time Distributed Computing, ISORC 2005. [S.l.]: IEEE, 2005.
[11] ECHTLE K, MOHAMED S. Clock synchronization issues in multi-cluster time-triggered networks[C]//Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance. Berlin Heidelberg: Springer, 2010: 39-61.
[12] KOPETZ H, GRUNSTEIDL G. TTP-A protocol for fault-tolerant real-time systems[J]. IEEE Computer, 1994, 27(1): 14-23.
[13] STEINER W. TTEthernet specification[S]. [S.l.]: TTTech Computertechnik AG, 2008.

编 辑 税 红