U.P.B. Sci. Bull., Series C, Vol. 71, Iss. 4, 2009 ISSN 1454-234x

THE ANALYSIS OF CAN AND ETHERNET IN DISTRIBUTED REAL-TIME SYSTEMS

Siegfried COJOCARU[1], Constantin RĂDOI[2], Ştefan STĂNCESCU[3]

Programarea în sistemele distribuite presupune programarea procesorului nodului local, programarea controlorului de comunicație şi programarea comunicației pe mediul comun de transmisie. O abordare sistematică este aceea de a separa programarea nodului local de programarea comunicației, astfel încât algoritmii implementați pentru programarea acestora să poată fi modificați independent unul de celalalt. În acest articol autorii analizează programarea comunicației şi influența acesteia asupra cerințelor sistemelor care funcționează în timp real pentru protocoalele CAN şi Ethernet.

Process scheduling in dedicated distributed system designs involves local node processor scheduling, communication controller programming and transmission scheduling. A systematic approach is to decouple node design from communication scheduling in a way that algorithms used to implement the solution can be changed independently. In this paper authors will analyze communication scheduling and its influence on real time requirements for CAN and Ethernet medium access protocol.

Keywords: controller area network, Ethernet, real-time systems, communication scheduling

1. Introduction

In a modern control system different functions are implemented in electronic control units (ECU), which cooperate in order to accomplish their task. Traditionally, a dedicated cable connects each ECU in the general design, but as the solution functionality increases significantly and more and more ECUs are needed to be connected, separate wiring was no longer a solution. An approach to manage such increasing complexity is to distribute the system functionality across several low costs microprocessors, which communicate via a shared medium. In this case new problems must be treated, such as how a receiver knows where a

[1] Eng., National Institute for Research and Development in Informatics, Bucharest, Romania, cojocaru_siegfried@yahoo.com

[2] Prof., Department of Applied Electronics and Information Engineering, University POLITEHNICA of Bucharest, Romania

[3] Prof., Department of Applied Electronics and Information Engineering, University POLITEHNICA of Bucharest, Romania

message starts and ends, with what rate the bits are sent and how the access to the bus is treated when more nodes wish to transmit simultaneously [1]. For this reasons, dedicated communication protocols must be developed. As many real time (RT) applications and hard real time systems (HRTS) are subject to timing constraints, developing network architectures and protocols capable to meet RT requirements are of great importance. In the last decades several different network protocols have been designed and implemented. But not all of them are capable to meet stringent requirements imposed by RT traffic.

Designing RT data and control networks relieve different goals to be taken into consideration. In common data systems, various amounts of data need to be transmitted at various speeds at unpredictable time instants. Data systems do not provide support for RT traffic. RT distributed control systems transmit control messages frequently, with short amounts of data exchanged between a large set of nodes. The ability to fulfill RT requirements represents the fundamental difference between control and data networks [2]. Industrial applications, as being essentially RT systems, use communication networks with some standardized real time deterministic Medium Access Control (MAC) layers, like Process Field Bus (PROFIBUS), Factory Instrumentation Protocol (FIP) , and Controller Area Network (CAN) [3].The most popular wired network architecture used for data transmission in common data systems, IEEE 802.3 , popularly called Ethernet, does not provide consistent support for RT traffic. The protocol uses the binary exponentially back-off (BEB) algorithm to solve collisions in a non-deterministic manner and does not support any message priorities. On a shared medium, it is possible, for a node that wishes to send a message, to wait a certain time until it gains the bus while other nodes are sending their messages. This time is known as blocking time and depends on the network protocol. It greatly affects the determinism and performance of a control network.

We discuss and analyze in the paper the capability of CAN and Ethernet network to meet the timing constraints imposed in RT systems.

2. Ethernet

In Ethernet network, a node access to the communication medium is made by the CSMA/CD (Carrier Sense Multiple Access with Collision Detect) technique. The CSMA/CD algorithm does not define a collision solving protocol of its own. The CSMA/CD protocol is specified in the IEEE 802.3 network standard and is described in detail in [4]. In CSMA/CD collision protocol, if two or more nodes collide, they back-off and try to retransmit message after randomly determined time periods. This random interval is determined by the standard BEB algorithm. CSMA/CD collision protocol is non-deterministic, so the Ethernet bus is not appropriate for real-time systems and totally not recommended to HRTS.

It is very difficult to exactly estimate blocking time delay for Ethernet. At a high level, the expected blocking time can be described by the following equation [5]:

$$E\{T_{block}\} = \sum_{k=1}^{16} E\{T_k\} + T_{resid}. \tag{3}$$

Where T_{resid} represents the residual time that is seen by node i until the network is idle, and $E\{T_k\}$ is the expected time of the *k*th collision.

3. CAN - Controller Area Network

CAN protocol was initially developed to be used for exchanging information between subassemblies in an automobile truck or vessel. Because of its low-cost implementation and built in error-detection scheme the protocol was successfully applied to other process control systems.

The CAN bus uses a CSMA/CD+AMP (Carrierer Sense Multiple Access with Collision Detection and Arbitration on Message Priority) access protocol. It means that whenever a node wants to use the CAN bus, it first needs to detect whether the bus is occupied or not. The node that wants to transmit a message waits until the bus is free and then start sending their dominant bit of start. During the arbitration process each node reads through its physical transceiver the bus logical value and compares it with the one sent by himself. If one node sent a dominant bit while the other sent a recessive bit, they will both read a dominant bit. The node, which sent the recessive bit, loses the arbitration, withdraws from the bus and turns into a receiver.

The blocking time in CAN T_{block}, is calculated in [6], by the following equation :

$$T_{block}^{(k)} = T_{resid} + \sum_{\forall j \in N_{hp}} \left\lceil \frac{T_{block}^{(k-1)} + T_{bit}}{T_{peri}^{(j)}} \right\rceil T_{tx}^{(j)}. \tag{1}$$

where

T_{resid} is residual time needed by the current node to finish transmitting,

N_{hp} is the set of nodes with higher priority than the waiting node,

$T_{peri}^{(j)}$ is the period of the *j*th message,

$T_{tx}^{(j)}$ is transmission time of node j on the network medium and

$\lceil x \rceil$ denotes the smallest integer number that is greater than x .

It is possible for a lower priority node to lose arbitration successively, while it is waiting to access the bus, against others node with higher priority who wish to send on the network. In this case, the total blocking time appear. The worst-case T_{resid} under a low traffic load is:

$$T_{resid} = \max_{\forall j \in N_{node}} T_{tx}^{(j)} \tag{2}$$

where N_{node} is the set of nodes on the network.

4. Case Studies

TrueTime [7] is a MATLAB/Simulink-based tool that facilitates simulation of the temporal behavior of a multitasking real-time kernel executing controller tasks. It can be used for simulation and research of dynamic RT networked control systems.[8] In order to examine CAN versus Ethernet RT characteristics, we used TrueTime to exercise and analyze experiments in a communication network with induced CAN or ETHERNET traffic. Network parameters in the simulation model are: the message period, message size, message offset, message priority and data transmission rate. In the experiments, each of a set of four nodes sends, at intervals of 10ms, messages of 8 data octets to the controller. Collisions may appear when some nodes wish to send simultaneously messages over the network. Consequently, due to arbitration process, different time delays will appear for different messages.

TrueTime trace the time in each experiment from the moment when a node wants to send a message until it reaches in the input buffer of the destination. Two of the network nodes will be created by configuring the TrueTime kernel block offered by the TrueTime library. The kernel is initialized using the function ttinitKernel to wich the corresponding parameters must be given. As the nodes send messages periodically, the function ttCreatePeriodicTask will be used to create a periodic task at each node. The functions ttCreateInterruptHandler and ttinitNetwork will be used to initialize the network. The function ttSendMsg will be correspondingly configured to send the message over the TrueTime network.

The other two nodes will be created using the built in block named ttSendMsg, as seen in the Fig. 4.1. These are configured through the Block Parameters dialog box. The ttSendMsg block has a Simulink trigger input port which can be configured to trigger on raising, falling, or either flanks. In this case a pulse generator is connected at this block and the trigger port is set on “either”.

As seen in the Fig. 4.1 a priority input port is also set to allocate message priority in the case when the protocol allows this (CAN). The simulation network is represented in the Fig. 4.1 below.

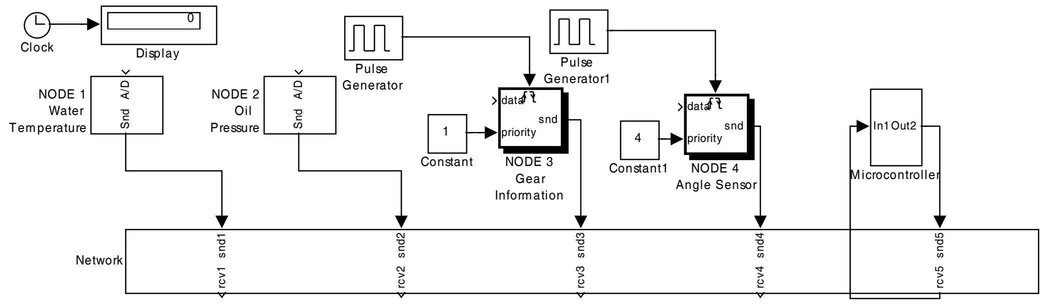

Fig. 4.1. Network model

We consider that the most important message, Gear_Information, is sent by node N3, followed by Oil_pressusre message send by node N2.

The first experiment is an Ethernet network simulation and analysis . The transmission speed is chosen 1Mbps for Ethernet and 250Kbps for CAN network. In the network schedule, the states of the network node are represented by the following level: high=sending, medium=waiting, low=idle. As can be seen in the Fig. 4.2 some messages loose arbitration in favor of others which are the first to gain the access to the network.

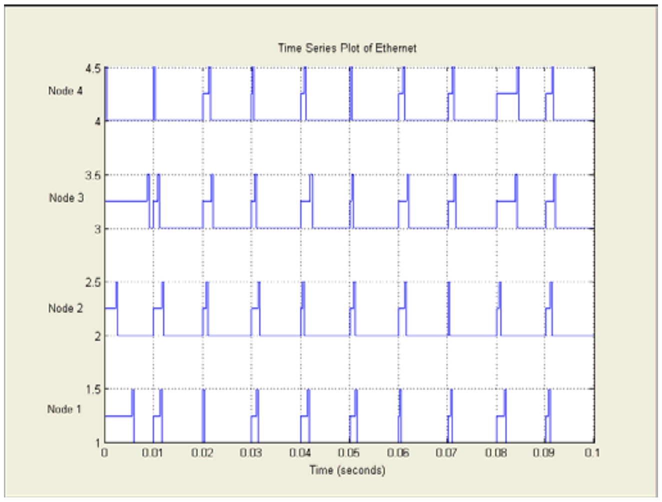

Fig. 4.2. Network scheduling for Ethernet

When trying to send its first message, node 3 (Gear_Information) has to wait, for example, 8.724 ms until he gets access to the bus and then transmit the message. In this case node 3 is the last to gain access to the bus. In the case of Ethernet, the nodes involved in collision wait an arbitrary time until they try to access the bus again. This leads to various time delays from the moment a node wishes to send until it gains access to the bus. Another delay for node 3 appears, for example, at the moment 0.03s, when sending its 4th message. This time, it is the second node to gain access to the bus, having to wait a much smaller time, 0.6ms, to access the network. On the other hand, node 2 (Oil_pressure) gains arbitration at moment 0.07 s and sends its 8th message immediately, having to wait instead any other time when it wants to access the network. To send its first message, it has to wait, for example, 2.284ms until he gets the bus. The time delay of a message represents the difference between the moment when a node is ready to send the message and the moment when it is received by the destination node. The variety of time delays of these messages illustrates the non-deterministic mechanism used by Ethernet protocol to regulate access to the bus. In RT systems it is important that the deadline for a message is guaranteed. One can notice that in this case the deadline for the Gear_Information and Oil_pressure cannot be guaranteed because many times they loose arbitration and have to wait until other messages are transmitted.

Keeping the same network configuration, the second experiment simulates and analyses a CAN network. CAN protocol offers the possibility to allocate priority to messages according to their importance.

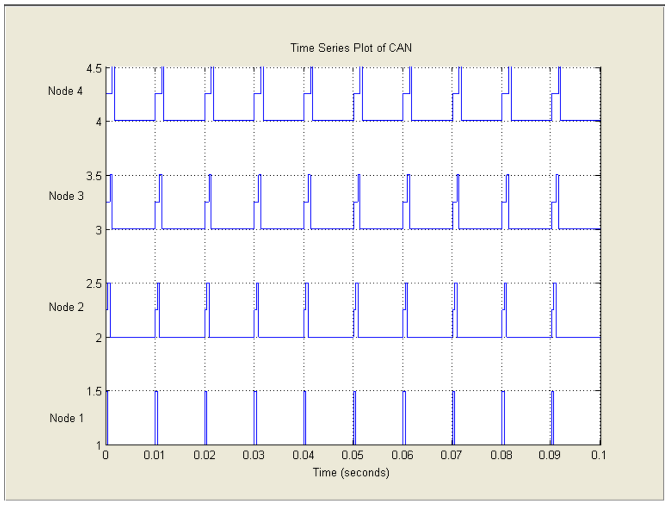

Fig 4.3. Network scheduling for CAN

Suppose we give node1, node2, node3 and node4 the priorities 1, 2, 3 and 4 respectively. As can be seen in the Fig. 4.3, node 3 (Gear_Information) always looses arbitration in favor of node 2 and 1. In the same way node 2, to which priority 2 is assigned, loses arbitration against node 1. It is to be noticed that CAN has a constant time delay which depends on the message priority.

For Ethernet network, however, the average time delay is not constant and it is difficult to predict even under low traffic conditions, that some messages must be rejected due to collisions, accordingly to BEB algorithm. The values of mean time delay calculated in table.1 illustrate the nondeterministic feature of time delay in Ethernet even when the network is not saturated. By comparison, the CAN network presents constant average time delay and 0 values for standard deviation. There is a linear trend for message time delay depending on the message priority. This is due to the priority based algorithm and the periodicity of messages which allow CAN to guarantee constant time delay for its messages.

Table 1

Simulation results for Ethernet and CAN

	Med. Ethernet [ms]	Ethernet Deviation [ms]	Med. CAN [ms]
Node1	1.6706	2.431454318	0.444
Node2	1.3184	1.024334711	0.888
Node3	2.6229	3.687007354	1.332
Node4	1.2702	1.840722385	1.776
Med	1.720525	2.245879692	1.110

As node 3 and node 2, respectively, send the most important messages in the network the design of the network can be improved by giving node 3 priority 1, node 1 priority 3, and choosing an offset for node 2 equal to 2ms.The simulation results indicate that better performances are obtained for node 3 and node 2. Running on a CAN network, node 3, which has the highest priority messages, will be the one to gain first access to the bus in any collision situation. By choosing a certain offset for node 2 it is possible to avoid collision with other messages. This offset depends on the size of messages to be transmitted, their period and the network transmission rate. Node 3, and node 2, with an offset of 2ms, request transmission at every 10 ms and don’t have to wait any longer to transmit their message. By comparison, node 4 has to wait for both node 3 to finish first its transmission and then for node 1. By configuring the network this way deadline can be guaranteed for message Gear_Information sent by node 3 and Oil pressure sent by node2.

§5. Conclusion

On a RT bus, response time of messages should be bounded. The Ethernet protocol uses the BEB algorithm to solve collisions in a non-deterministic manner and does not support any message prioritization. Ethernet is not suited for traffic with RT requirements. When the node has to send the highest priority message, the CAN arbitration mechanism allows it to immediately access the bus and transmit without any delay. In CAN protocol, the worst case response time for highest priority message is deterministic, and priorities can be allocated to messages according to their importance in order to meet their timing requirements. Choosing a certain offset for messages, collision may be avoided in some extent. For control systems with short and/or prioritized messages, CAN offers a better performance.

R E F E R E N C E S

[1] *Stallings William*, Data and computer communication, ISBN 0-13-084370-9, 2000
[2] *R.S. Raji*, Smart networks for control, IEEE Spectrum, **vol. 31**, no. 6, pp. 49-55, June 1994
[3] *Etschberger Konrad*, Controller Area Network (CAN) Grundlagen, Protokolle, Bausteine, Anwendungen, Dritte Auflage. Hanser Verlag, München Wien, 2002
[4] *A.S. Tanenbaum*, Computer Networks, Upper Saddle River, NJ, Prentice-Hall Inc., third edition, 1996
[5] *F.-L. Lian, J.R. Moyne, D.M. Tilbury*, Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet, Technical Report: UM-MEAM-99-02, 2001
[6] *K. Tindell, A. Burns, A.J. Wellings*, Calculating Controller Area Network (CAN)message response times, Control Engineering Practice, **vol. 3**, no. 8, pp. 1163-1169,Aug. 1995.
[7] *M. Ohlin, D. Henriksson, A. Cervin*, TrueTime 1.5—Reference manual, Department of Automatic Control, Lund Institute of Technology, January 2007
[8] *Andersson, Martin, Dan Henriksson, Anton Cervin, Karl-Erik Årzén,* (2005) Simulation of wireless networked control systems. In: *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*. Seville, Spain