

FTT-CAN: A NETWORK-CENTRIC APPROACH FOR CAN-BASED DISTRIBUTED SYSTEMS

Luís Almeida, José A. Fonseca
{lda,jaf}@det.ua.pt

DET - IEETA
Universidade de Aveiro
P-3810-193 Aveiro, Portugal

Abstract: The requirement for flexible operation is becoming increasingly important in many distributed computer control systems. This paper briefly presents the FTT-CAN protocol (Flexible Time-Triggered communication on Controller Area Network) which allows for flexible operation, i.e. dynamic communication requirements, under timeliness guarantees. Then, the paper discusses a particular approach to the global management of such distributed systems named network-centric. This is based on the synchronisation of the distributed tasks in remote nodes with the network traffic. Thus, on-line changes in the rates and/or relative phasing of the network traffic can be automatically applied to the tasks that use it, in all the remote nodes in the distributed system. This approach is fully supported by the FTT-CAN protocol. *Copyright © 2000 IFAC*

Keywords: Real-time communication, Real-time systems, Communication protocols, Distributed computer control systems, Fieldbus.

1. INTRODUCTION

The growing importance of flexibility in modern industrial real-time systems comes from the need to make those systems capable of operating in dynamic environments, i.e. where a complete specification is not possible and where operational requirements may change during system lifetime. However, flexibility must be achieved without jeopardizing the understandability and predictability of the system temporal behaviour allowing safe and predictable upgrades with negligible downtime.

The time-triggered communication paradigm is particularly well suited to support such predictable temporal behaviour since the release times for any message transmission, given a set of communication requirements, can be pre-determined. Moreover, this paradigm allows the use of relative phase control to reduce jitter in periodic message transmissions and also to support composability with respect to the temporal behaviour (Kopetz, 1997), i.e. the temporal

behaviour of a given subsystem is not affected when it is integrated in the whole system. These characteristics make this paradigm particularly well adapted to control applications that typically require regular transmission of state data with low, or bounded, jitter (e.g. motion control, engine control, temperature control, position control).

Another consequence of relative phase control is that higher network utilisation under guaranteed timeliness can normally be achieved when comparing with systems based on event-triggered communication. In these ones, the asynchronous nature of events forces to consider a worst-case situation where all events occur simultaneously. In time-triggered systems, the relative phase control can be used to reduce the number of messages that are ready for transmission at any moment, thus reducing the network demand in the worst-case scenario.

One of the problems with the time-triggered approach is that it is normally associated to static off-line scheduling, thus with low operational flexibility. To solve this problem, the authors have presented in

(Almeida *et al.*, 1999a) a planning scheduler that uses a dynamic table-based concept that allows introducing the desired level of operational flexibility in those systems. Furthermore, the planning scheduler uses a centralised scheduling approach that facilitates the dynamic management of the communication requirements.

Some preliminary experiments to explore the application of the time-triggered paradigm based on the planning scheduler in the Controller Area Network – CAN (Bosch, 1991) have been presented in (Almeida *et al.*, 1999b). These led to the development of a CAN-based protocol named FTT-CAN (Flexible Time-Triggered communication on CAN). The next section briefly describes this protocol. One of its features is the support for dynamic communication requirements. Furthermore, it delivers communication services synchronised with the network traffic that allow the tasks running in remote nodes to automatically follow the changes performed on the traffic properties. This approach to the control of distributed applications in fieldbus-based systems is called the network-centric approach and it is explained in section 3. Section 4 presents some conclusions.

2. FTT-CAN: A FLEXIBLE TIME-TRIGGERED PROTOCOL FOR CAN

The FTT-CAN protocol aims at combining some level of operational flexibility with the time-triggered communication paradigm. The result is an autonomously controlled communication system that, nevertheless, allows on-line reconfiguration under guaranteed system timeliness. The main advantage of this system is that it supports dynamic expression of communication requirements with limited communication and processing overhead. Hence, it can be easily implemented without any special hardware support, using low processing power microcontrollers (an experimental set-up has been built (Almeida *et al.*, 1999b) based on Philips 80C592 microcontrollers clocked at 11MHz, using a bus transmission rate of 125Kbit/s).

Two types of messages are considered: synchronous and asynchronous. The former ones are transmitted periodically and automatically by the communication system. A requirements table (M_s) describes the periods (P), deadlines (D), sizes (S), relative phasing (Ph) and priorities (Pr) of all these N_{M_s} messages:

$$M_s \equiv \{ m_i = m_i(P_i, D_i, S_i, Ph_i, Pr_i), i=1..N_{M_s} \}$$

Synchronous messages are guaranteed to meet the respective deadlines by using an appropriate on-line schedulability analysis. In what concerns asynchronous messages, they are handled by the communication system in an event-triggered fashion. No timeliness guarantees are given in this case and these messages are handled according to a best-effort

approach. However, the protocol assures a clear separation between synchronous and asynchronous messages so that there is no negative interference of the latter ones on the timeliness of the former ones. The protocol assigns two separate ranges of identifiers for each of these two types of messages. The synchronous ones get the higher priority identifiers whereas the asynchronous ones get the lower. Also, the number of synchronous messages (N_{M_s}) is limited to 63 in the current version, and the number of asynchronous messages is limited only by the availability of CAN identifiers in the respective range (at least half of the CAN, version A, full range).

2.1 Centralised scheduling with distributed arbitration

The main advantage of using centralised scheduling resides in the fact that all the synchronous communication requirements are localised in one node, the master. Hence, when comparing with a time-triggered TDMA-based distributed approach, e.g. TTP (Kopetz and Grünsteidl, 1994), it is easier to perform changes on those requirements such as adjusting the period, priority or relative phasing of a message, or even adding/deleting messages. On the other hand, centralised scheduling is normally associated with centralised access control, e.g. the WorldFIP fieldbus (CENELEC, 1996). However, in CAN there is a well-defined deterministic distributed arbitration protocol that can be used to reduce the typically large communication overhead of centralised access control (Almeida *et al.*, 1999a).

Hence, such as in WorldFIP (in the synchronous mode), bus time is broken down in fixed duration slots called *elementary cycles* (EC) within which bus transactions take place. All periods and deadlines are then expressed as integer multiples of the EC duration defined at start-up (in the experimental set-up referred before, the EC duration was set to 8.9ms). The master node instead of sending one identification message for every transaction (as in WorldFIP), it sends only one message at the beginning of each EC. Such message, the *EC trigger message*, carries the identification of all transactions that must be carried out during one EC (fig. 1-a). The involved producers respond to the call trying to transmit their messages concurrently. Contention on the bus is left for the CAN native arbitration to sort out (fig. 1-b). The EC duration is taken into account by the centralised scheduler so that all transactions scheduled for the same EC are guaranteed to fit in. This aspect allows guaranteeing a high regularity for the EC trigger message, which then, can be used as a system wide precise synchronisation signal. Besides, the protocol reserves a range of high priority identifiers for the master messages. In the present implementation there are only two types of master messages, the EC trigger message and the M_s download message (see further). However, several

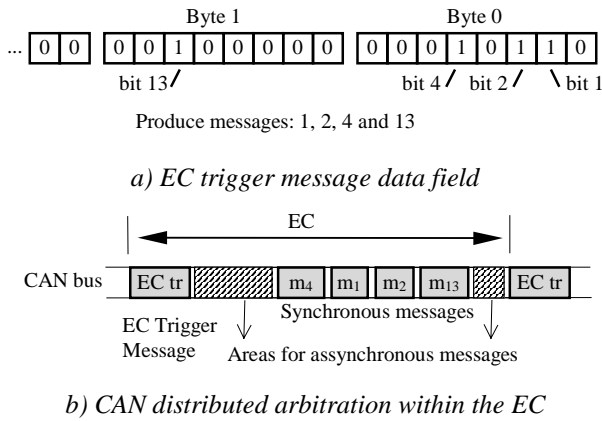


Fig. 1. Centralised scheduling with low overhead.

different types are possible, for example to communicate particular system configuration changes to the nodes.

Notice that, in this protocol, the priority of each synchronous message is a parameter in the Ms table, which is separated from the respective CAN identifier. The priority is used by the scheduler to establish the ECs in which the message will be transmitted. The identifier is used by the CAN native arbitration to decide when within the EC will the message be transmitted. The transaction timeliness in terms of meeting the deadline is controlled by the priority but, nevertheless, the identifier influences the transmission jitter within the EC.

2.2- The asynchronous messaging system

The FTT protocol can take advantage of the bus-idle periods in the EC, i.e. not used by the synchronous messaging system (fig. 1-b) to allow event-triggered communication. This communication uses asynchronous messages (asynchronous because the instants in time in which they become ready to be transmitted are not synchronised with the scheduler activity). The FTT-CAN protocol ensures that asynchronous messages are transmitted in a way that does not jeopardise the temporal behaviour of the synchronous system. This is achieved either by enforcing that asynchronous transmissions occur only in the allowed time slots within each EC or by considering the possible blocking effects of those transmissions on the synchronous ones when building the schedules.

Nevertheless, for the purpose of this paper, the asynchronous messaging system will not be further considered.

2.3- Support for temporal accuracy

A real-time distributed system must always be aware of the environment state. For this purpose, it maintains a distributed data structure that mirrors such state, i.e. the real-time database (Kopetz, 1997). Each item within the real-time database has a limited time validity due to the dynamics of the environment.

When an item is within its validity interval it is said to be temporally accurate.

It is of major importance to the applications using producer-consumer interactions (Peraldi and Decotignie, 1995) to know whether the value of any consumed state variable is temporally accurate. The FTT-CAN protocol is intended to support temporal accuracy information for synchronous traffic in a way somewhat similar to WorldFIP but more simplified. Notice that this type of traffic is normally used to convey the updated value of system variables that are required across the system. Whenever a variable is consumed, the statuses of two boolean time properties are delivered together: the *refreshness* and the *promptness* statuses. The refreshness status is generated at the producer node to indicate whether the produced value started to be transmitted within its refreshness interval. This boolean indication is codified within one bit of the identifier of the respective message and transmitted together with it. On the consumer side, a timer is set with a pre-configured value, the promptness interval, as soon as the respective message is received. The promptness status indicates, then, whether the consumer task has used the new value within the promptness window (fig. 2). The sum of the refreshness and promptness windows plus the message transmission time should equal the validity interval of the respective state variable.

In simple systems, the timer unit may not be sufficiently versatile to support the refreshness and promptness information as described above. In that case, the FTT-CAN protocol may use a simplified version of the above scheme that does not require timers. The refreshness window is set equal to the production period of each message and the promptness information is replaced by a *new data* boolean indication that informs the consumer that it is consuming either a new or old (i.e. already used) value.

2.4- Flexibility using the planning scheduler

To achieve flexibility in what concerns on-line changes to the requirements table Ms, a dynamic scheduler must be used. However, dynamic scheduling brings along a significant run-time overhead, which, on low-processing power microcontrollers, would necessarily cause an impediment to an efficient network bandwidth

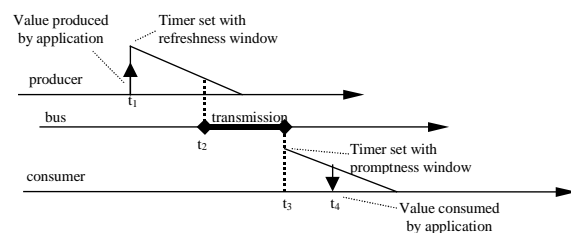


Fig. 2. Temporal accuracy information.

utilization. Thus, in order to reduce the run-time overhead, the FTT protocol relies on a dynamic table-based scheduler known as *planning scheduler* (Almeida *et al.*, 1999a).

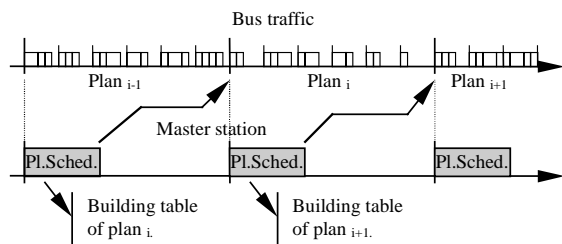
This scheduler operates on the requirements table Ms producing a schedule table for a fixed duration period of time called a *plan*. The plan duration is independent of the messages' periods and so the plan is not cyclic in general, needing to be rebuilt at the end of each scan (fig. 3-a). Thus, changes to the Ms table can be accounted for when building the next plan.

Two different activities can be identified within the master node: scheduling and dispatching. In order to effectively reduce the run-time overhead, these activities are overlapped in time. This leads to the use of two plan tables: while the dispatcher scans the current plan and initiates the respective transactions, the scheduler builds the next plan (fig. 3-b). Experiments described in (Almeida *et al.*, 1999b) showed that increasing the plan duration from 1 to 20ECs (178ms with an EC of 8.9ms) caused the run-time overhead to be reduced by a factor close to 3.

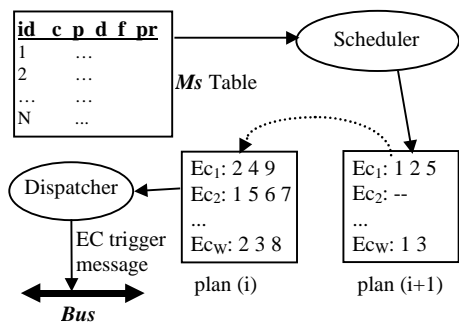
Also, to guarantee a higher reduction in run-time overhead, the scheduler must use a static priorities-based criterion to build the table. The messages' priorities defined in the Ms table are used. The relationship between period or deadline and priority is left for the user to define.

2.5- On-line admission control

One of the features of the FTT protocol is that on-line changes to the Ms table are only allowed if a



a) operation



b) functional decomposition

Fig. 3. The planning scheduler.

schedulability analysis guarantees the overall timeliness of the synchronous messaging system. Such analysis has, thus, to be carried out at run-time. This analysis can be executed in the master node together with the planning scheduler. However, in systems based on low processing-power micro-controllers, the analysis might represent an excessive load. Therefore, in order to avoid the consequent extra load on the master node, the analysis can be moved to a different node, the operator console, that assists the master in all the issues related to *on-line admission control*.

In (Almeida and Fonseca, 1998) the authors identify two distinct levels of system reactivity to changes on the communication requirements. A more demanding level is associated to change requests coming from tasks in order to cope with sudden changes in the environment (e.g. a line-tracking robot might increase the sampling of the line position whenever a sharp curve is detected). A less demanding level is associated to change requests coming from a human operator (e.g. the addition of a new sensor might require a new synchronous message, the update of a controller might require a different set of messages). While in the former case reaction times in the order of few milliseconds might be required, in the latter case a reaction time in the order of a few seconds is normally tolerable.

Essentially due to run-time overhead considerations, the present version of the FTT protocol has been designed to support changes issued by an operator, only. Hence, any requests for changes to the communication requirements table Ms come from a single node, the operator console. This node is, thus, responsible for three tasks: the operator interface, the admission control and the master's Ms table management (fig. 4). The interface is performed with the help of a dumb terminal connected to the node via RS232. This interface gives access to the Ms table management functions such as listing, adding/deleting messages, changing messages' parameters.

The admission control includes two different tests. The first one performs a response time-based schedulability analysis similar to the one proposed in (Almeida and Fonseca, 1999), taking into account the

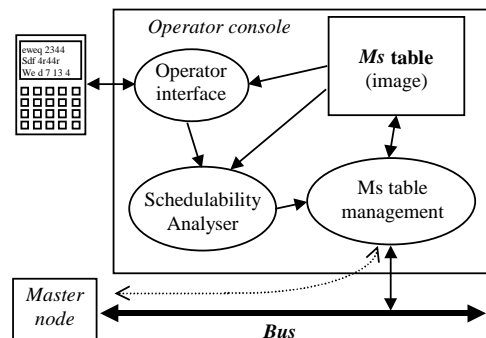


Fig. 4. The operator console.

messages' fixed priorities. The second test is particularly important when using low-processing power micro-controllers in the master node, otherwise it can be eliminated. In this case a model of the scheduler execution time is used to check whether the requested changes do not cause the scheduler in the master node to exceed its CPU load budget. If both tests return a positive result then the requested change is accepted.

The master's Ms table management is achieved with a special purpose protocol between the operator console and the master node. The information concerning change requests is transmitted to the master using a synchronous message produced by the operator console, i.e. the *operator message*. On the other way, the master acknowledges each change using a special asynchronous message, i.e. the *Ms download message*, transmitted in one asynchronous window when required, only. This protocol also supports the downloading of the whole requirements table from the master to the console. This feature is particularly important to support a dynamic connection of the operator console, i.e. this node can be connected to the network only at system start-up or when the operator wishes to perform any change, or verification, on the Ms table. This detail also adds to the operational flexibility of the system.

2.6- Higher layer services

In section 2.3 it was already referred that Producer-Consumer (PC) interactions are particularly useful for communication based on periodic streams, as commonly found in control applications. Client-Server (CS) interactions are better suited to support reliable sporadic data exchanges. The FTT-CAN higher layer delivers two different types of communication services, each following one of those co-operation models. The PC-based services use the time-triggered, i.e. synchronous, traffic whilst CS-based services use the event-triggered, i.e. asynchronous, traffic.

The PC-based services use the status information on temporal accuracy. Furthermore, these services include an option to synchronise the application software with the transmission/reception of an associated message. By using this option, it is possible to force the rate of the cyclic execution of producers or consumers in remote nodes to match the rate of transmission of a specified message on the network. This feature is the basis for the network-centric approach explained in the following section.

The envisaged CS-based services include the possibility to send data with or without acknowledgement as well as to send and request data, and request data on response.

Apart from the communication services, the FTT-CAN higher layer also includes services for managing the synchronous communication requirements such as

add new message, remove existing message and update message attributes. These services directly interact with the messages' admission control.

3. THE NETWORK-CENTRIC APPROACH

The network-centric approach considers the fieldbus system within a distributed computer controlled system as an independent fundamental element. Its distinguishing property is that it interfaces with all the nodes and thus, it is particularly well positioned to support the global control of the whole distributed system. In fact, any nodes in the distributed system can only interact with each other through the respective network interfaces, using the fieldbus communication services. Therefore, the fieldbus appears as the only common infrastructure when considering all the distributed activities being executed over the whole distributed system. In one side there are the nodes that send messages (possibly in the course of more complex bus transactions). On the other side there are the nodes that receive those messages. The fieldbus is at the centre, supporting all message transmissions, from senders to receivers.

The central positioning of the fieldbus becomes particularly important when considering the operational flexibility of the distributed system as a whole. A flexible system must allow on-line configuration changes, such as adding or removing nodes as well as changing operational parameters either of tasks and messages. For example, consider the following scenario: In a process industry, the control engineer establishes the right sampling rates for certain sensors and, consequently, the rate of transmission of those samples on the network. However, after putting the system on, part of the process shows an undesired oscillation, which is due to the under-estimation of some inertial factors. Among other things, the problem requires some sampling rates to be increased. In a flexible system, it would be possible for the control engineer to adjust the sampling periods on-line, including the transmission rates of the respective samples over the network. The adjustment should, nevertheless, be carried out under the supervision of an adequate admission control in order to assure the continuous timely behaviour of the system.

In a distributed system using autonomous time-triggered communication, a given producer node A executes a task T_A that continuously samples some state variables of the environment and produces the respective values (i.e. makes them available in the network interface). The communication system, in turn, cyclically transmits a message M_A to broadcast those values. In the consumer node(s) B a task T_B continuously consumes those values and uses them to generate some output. Notice that there is no direct relationship between the rates at which the three activities, T_A , T_B and the transmission of M_A are

executed. Therefore, the flow of information from node A to node B progresses at a rate determined by the slowest of the three.

An efficient approach in what concerns CPU and network utilisation is to force the three rates to be equal using some synchronisation mechanisms. For example, this can be achieved by forcing the production / consumption services to wait for the respective message transmission / reception. In this case, all the distributed cyclic activities in the system execute at a rate imposed by the transmission of messages. Hence, by controlling the communication requirements specified in the communication system, an operator can easily control the rates of the different information flows and associated activities being processed in the system. Furthermore, this level of control can also be applied to the relative phase of those flows, a feature that is not possible with event-triggered communication, due to the asynchronous nature of events.

This dynamic and synchronised approach to the control of the information flows and associated activities in a distributed system is named the network-centric approach. The FTT-CAN is an example of a protocol that follows this approach. In fact, the higher layer services based on the producer-consumers model (see 2.6) include an option for synchronising with the respective message transfer. By making an adequate use of that option, these services allow to develop distributed applications, which support a network-centric control of the respective information flows and activities. Through the operator console, an operator can establish on-line either the rates as well as the relative phasing of the flows of information of a running application. This feature can be used to support the desired system operational flexibility, e.g. as required in the scenario presented in the beginning of this section.

4. CONCLUSION

This paper discusses the advantages of using time-triggered communication instead of event-triggered one. The major feature associated to that sort of communication is that it allows relative phase control among the several streams of messages. Based on that feature, time-triggered communication supports composability with respect to the temporal behaviour and it also supports some level of jitter control. However, it is typically associated to static off-line scheduling, thus having low operational flexibility.

In this paper, the FTT-CAN protocol (Flexible Time-Triggered communication on CAN) is briefly presented. It combines time-triggered communication with flexibility under guaranteed timeliness. It uses centralised scheduling but makes use of the native CAN distributed arbitration to achieve a reduced communication overhead.

The paper then focuses on a new perspective towards the control of distributed applications in fieldbus-based systems. This is called the network-centric perspective according to which the flows of information and the associated activities that are part of a given application are controlled directly from a single point either for their rate as well as relative phasing. This perspective is based on considering the fieldbus as the pivot element in the distributed system that has to support all the interactions among nodes. The FTT-CAN protocol follows such a network-centric approach and thus, it facilitates the on-line control of distributed applications.

REFERENCES

- Bosch (1991). *Controller Area Network (CAN) specification – version 2.0, part A*. Bosch GmbH, Germany.
- Kopetz, H. and G. Grünsteidl (1994). TTP - A Protocol for Fault-Tolerant Real-Time Systems. *IEEE Computer*, **27**(1).
- Kopetz, H. (1997). *Real-Time Systems Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers.
- CENELEC (1996). European standard EN 50170. *Fieldbus: Vol.1: P-Net; Vol.2: PROFIBUS; Vol.3: WorldFIP*. CENELEC, European Com. for Electrotechnical Standardisation.
- Peraldi, M.A. and J.D. Decotignie (1995). Combining Real-Time Features of Local Area Networks FIP and CAN. *Proc. of ICC'95 (2nd Int. CAN Conf.)*. CiA – CAN in Automation.
- Almeida, L. and J.A. Fonseca (1999). Schedulability Analysis in the FIP Fieldbus Accounting for Inserted Idle-Time. *Proc. of WIP session of RTS'99 (1st EUROMICRO Conf. on Real-Time Systems)*, York, UK.
- Almeida, L. and J.A. Fonseca (1998). The Planning Scheduler: Compromising between Operational Flexibility and Run-Time Overhead. *Proc. of INCOM'98 (IFAC Int. Symposium on Information Control in Manufacturing)*, Nancy/Metz, France. June.
- Almeida, L., R. Pasadas and J.A. Fonseca (1999a). Using a planning scheduler to improve flexibility in real-time fieldbus networks. *IFAC Control Engineering Practice*, **7**: 101-108.
- Almeida, L., J.A. Fonseca and P. Fonseca (1999b). A Flexible Time-Triggered Communication System Based on the Controller Area Network: Experimental Results. *Proc. of FeT'99 (Int. Symp. on Fieldbus Technology)*, Magdeburg, Germany.