

## **Different schedules Proposals for Successive Transmission for the General Purpose Network Controller**

**Abdurazzag Ali Aburas and Manal Ibrahim Al Fellah**

*International Islamic University Malaysia, Dept. of Electrical and Computer  
Engineering and Dept. of Information & Communication  
Technology Kuala Lumpur, Malaysia  
E-mail : manalfalleh@iiu.edu.my  
E-mail : aburas06@iiu.edu.my*

### **Abstract**

General Purpose Controller Network Architecture is an experimental network architecture designed for interconnecting nodes in distributed embedded applications. It provides deterministic access to the network medium to support distributed hard real-time application components. General Purpose Controller Network Architecture uses a token bus to transfer fixed sized frames. Token monitor issues an explicit token that contains the schedule using which the nodes are allotted medium access. The 37 byte frame structure includes a 5 byte header and a 32 byte payload. Payload within the token carries single byte addresses of the nodes on the network sorted in the order they are granted access to the medium. Only the node in possession of the token is granted medium access, and this node subsequently transmits the token to the node scheduled next in the token payload. Thus, the original General Purpose Controller Network Architecture allows only up to 32 nodes to be present on the network. This paper suggests three optimizations to the original General Purpose Controller Network Architecture concept. Primarily, the use of virtual tokens reduces the token overhead. Nodes can dynamically request bandwidth allocation to transmit periodic or sporadic messages. Finally, categorization of nodes as talkers and listeners and use of different schedules for successive transmission cycles increases the number of nodes on the General Purpose Controller Network Architecture bus.

**Keywords:** Distributed embedded systems, real-time communications, computer networks, medium access control, MAC.

## **Introduction**

Messages in real-time communication are often categorized into synchronous and asynchronous messages. Synchronous messages are produced and consumed on a continuous basis. Such traffic includes periodic and sporadic messages that require some guarantee for on-time delivery. Asynchronous or periodic messages have soft timing constraints and are expected to be delivered on the best effort basis [1].

Significant research has been conducted in scheduling nodes in distributed embedded systems for transmission such that messages arrive at their destinations within the specified deadlines. One of the most notable developments in this field has been the specification of the Controller Area Network (CAN) [2]. CAN uses a priority based medium access control (MAC) method referred to as the CSMA/AMP (Carrier Sense Multiple Access/Arbitration on Message Priority). Briefly, each message has an identifier that is based on the message contents. Lower identifiers are granted higher priorities as the nodes are connected to a bus in a wired-AND logic. CAN frames carry 8 byte message per frame thereby making it suitable only to communicate instrument measurements, machine status, actuator commands and machine settings. Furthermore, lower priority messages may suffer indeterminable latencies in the presence of high priority messages [3].

Time Triggered CAN (TT-CAN) [4] has been developed to further optimize message scheduling in CAN. In TT-CAN, the exchange of messages is controlled by the temporal progression of time. Messages are scheduled at predefined points relative to the time of transmission of a reference message. The reference message may be transmitted either periodically (time triggered mode) or on the occurrence of a particular event (event triggered mode). Nodes then communicate using a static schedule. TT-CAN provides two types of transmission windows, the arbitrating time windows and the exclusive time windows. In the arbitrating time windows, several nodes can compete for the bus on the basis of message priority. However, only one message from a specific network node may be transmitted during an exclusive time window [4].

Another variation of CAN known as FTT-CAN (Flexible Time Triggered CAN) also relies on centralized scheduling of messages but makes use of the CAN native distributed arbitration to reduce communication overheads [5]. FTT-CAN supports both time-triggered and event-triggered traffic with temporal isolation being achieved by means of a double phase elementary cycle. The former type of traffic is scheduled in the synchronous phase of the cycle whereas the latter type of traffic competes for bus access in asynchronous phase. Therefore, the synchronous phase of the elementary cycle can be used to schedule periodic messages whereas periodic and sporadic messages can be transmitted during the asynchronous phase with the protocol ensuring temporal isolation between the two phases [5].

Several token passing bus networks have also been developed to provide nodes in distributed embedded and control systems with deterministic access to the medium. IEEE 802.4 Token Bus was developed to connect nodes in a manufacturing plant [6]. Each node has access to the bus after receiving a token. The node in possession of the token transmits the token to the next node if it has no more data to transmit or its time for the possession of the token has expired.

ControlNet provides an interesting variation to token passing bus networks. It adopts an implicit token passing mechanism. The token to the next node is embedded in the last data frame that a node currently in possession of the token transmits. This embedded token is much smaller in size and therefore the token passing overhead of ControlNet is considerably less. Furthermore, ControlNet divides the bus cycle into three parts. During the scheduled part of the bus cycle, nodes can transmit time-critical messages by obtaining an implicit token. The unscheduled part of the bus cycle is shared by the nodes to transmit non-time-critical messages. Guard band is used to transmit synchronization and maintenance messages. ControlNet provides higher data coding efficiency than CAN and can also be used to communicate all types of traffic between nodes [3].

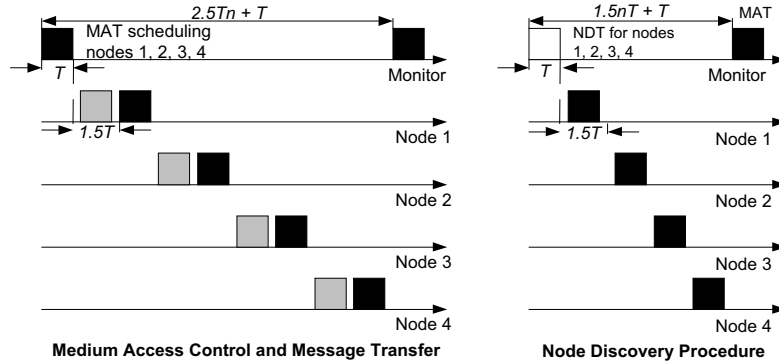
Final year students of College of Aeronautical Engineering (CAE) at the National University of Sciences and Technology (NUST) in Pakistan developed the General Purpose Controller Network Architecture. This architecture employs a token passing bus and optimizes frame delineation and data coding by using a fixed sized frame with a small header. Primarily this architecture was developed only to communicate periodic messages between nodes (up to 32 only on one network segment) [7]. However, several optimizations within this architecture are being considered, some of which shall be discussed in this paper.

Subsequent sections of this paper briefly describe the controller and highlight its limitations. Based on these limitations, several schedules are suggested before concluding this paper.

### **The General Purpose Controller Network Architecture**

Controller is a token bus based layer-2 network protocol for distributed embedded systems. The primary objective of developing this protocol was to provide the nodes in distributed embedded systems deterministic access to the network while maintaining simplicity and efficiency. Deterministic access to the network is accomplished through token passing. Fixed length frames allow simple and efficient frame delineation [7].

Controller uses two types of tokens, the Medium Access Token (MAT) and the Node Discovery Token (NDT) [7]. MAT is a directed token that is passed from node to node based on a schedule prepared by the monitor node. The monitor node embeds this schedule in the payload of the MAT it generates. The subsequent nodes use this schedule to transmit the token to the next node in the schedule after transmitting their frames. NDT is a broadcast token that is used in the Node Discovery Procedure to determine if nodes that did not respond in the previous transmission cycle (dead nodes) are alive and should be scheduled in the next transmission cycle. NDT carries the list of addresses of dead nodes in the payload. For each transmission of NDT, the nodes mentioned in the payload of NDT respond in the subsequent time intervals in the order specified in the payload of NDT. Each time interval is the time required to transmit a 37 byte controller frame. These time intervals include guard times (bands) to accommodate jitters in frame transmissions [7]. Figure 1, shows the different token modes of controller.



**Figure 1:** Different Token Modes

It should be noted that the Node Discovery Procedure is carried out at the end of each transmission cycle if during that transmission cycle one or more nodes failed to respond to the token transmitted to them or some dead nodes exist from the previous transmission cycles.

### The Basic Limitations

Controller provides the advantages of simple and efficient frame delineation, deterministic access to the medium and a payload size sufficient to allow most high-level messages to be communicated between nodes without message fragmentation. However, the basic controller suffers from several limitations. These limitations are highlighted in the following paragraphs.

### Token Passing Overhead

Controller uses explicit token passing. MAT and NDT are complete controller frames. Explicit token passing and Node Discovery Procedure, therefore, occupy substantial proportion of the transmission cycle. From Figure 1, it can be seen that the transmission cycle is  $2.5nT+T$  where  $n$  is the number of nodes on the bus and  $T$  is the time required to transmit a controller frame on the bus. Similarly the time occupied by all the MATs is  $nT+T$ . Therefore, the overhead associated with MATs is given by the following expression,

$$MAT\ Overhead = \frac{n+1}{2.5n+1} \quad (1)$$

The time consumed by the Node Discovery Procedure is  $1.5nT+T$  where  $n$  is the number of nodes considered dead in the previous transmission cycle or the total number of nodes on the bus at initialization. If  $k$  nodes are found dead in a specific transmission cycle, the subsequent node discovery procedure is considered a part of that transmission cycle for the purpose of analysis. The total transmission cycle is extended to  $(2.5n+1.5k+2)T$  whereas the time on the bus when only tokens are transmitted increases to  $(1.5k+n+2)T$ . The overhead in this case is given by the following expression,

$$\frac{1.5k+n+0.5j+2}{2.5n+1.5k-j+2} \qquad \frac{1.5k+n+2}{2.5n+1.5k+2}$$

Combined MAT and NDT Overhead = (2)

Combining (1) and (2) results in the complete expression for token passing overhead for controller,

$$\text{Token Passing Overhead} = \begin{cases} \frac{n+1}{2.5n+1} & k=0 \\ \frac{1.5k+n+2}{2.5n+1.5k+2} & 0 < k \leq n \end{cases} \quad (3)$$

Figure 2, shows a graphical visualization of the overhead from (3) with respect to the number of nodes on the network and the number of nodes that did not respond in the previous transmission cycle, i.e., dead nodes. For example, the series referred to as 0 in the legend specifies that there were no dead nodes detected in the previous transmission cycle and the token passing overhead is calculated for different number of nodes on the network. This graph clearly represents the substantial token passing overhead in controller.

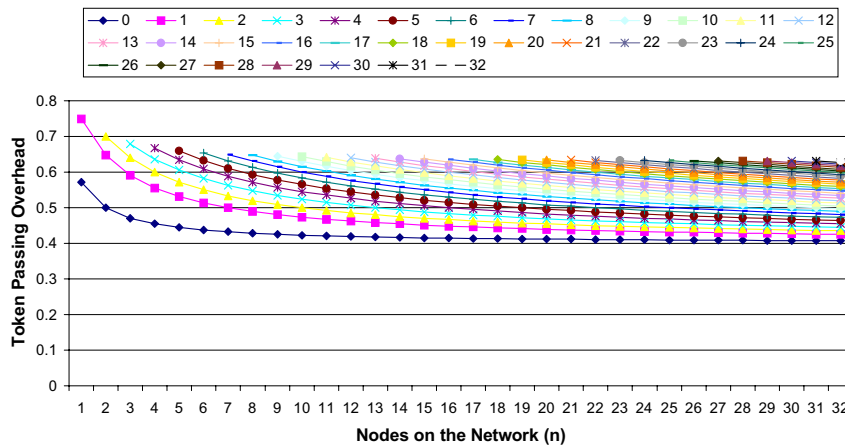


Figure 2: Token Passing Overhead in controller

Expression (3) does not take into consideration previously detected dead nodes. If there are  $j$  confirmed dead nodes before the start of a transmission cycle, only  $m$  nodes will be scheduled in that cycle, where  $m=n-j$ . If during this transmission cycle  $k$  dead nodes are discovered ( $0 < k \leq n$ ), then the total time when tokens are transmitted is calculated by  $1.5kT+0.5jT+nT+2T$ . The total length of the transmission cycle is  $2.5nT+1.5kT-jT+2T$ . The general expression for the overhead in the presence of dead nodes transforms to the following,

$$\text{Token Overhead with Dead Nodes} = \frac{1.5k+n+0.5j+2}{n+1.5k-j+2} \quad (4)$$

These results in a further increase in the overhead associated with tokens on controller.

### **Fixed Transmission Scheduling**

Controller in its basic form provides only fixed transmission scheduling. Monitor contains the schedule for the transmitters. This schedule is provided to each node via the MAT. Nodes that have a higher frequency of messages to be transmitted can be scheduled a multiple number of times in the transmission cycle. Therefore, controller is optimized only for the communication of periodic messages. Periodic and sporadic messages from applications can cause messages to accumulate in the transmission buffers and can delay the periodic messages substantially causing the entire system to malfunction. There are no means of scheduling nodes to transmit periodic or sporadic information. The only variation in scheduling is the removal of previously detected dead nodes from the subsequent schedules or to reinsert them when they are detected as alive by the Node Discovery Procedure.

### **Limited Number of Nodes per Segment**

As mentioned earlier, controller transmission schedule is maintained at the monitor. This schedule is transmitted to every node through the MAT as an ordered list of node addresses within the frame payload. Each node receiving the MAT can determine the identity of the node to which the MAT should be transmitted next. Node addresses are byte wide and therefore only a 32-node schedule can be specified in the MAT. This, coupled with static single cycle transmission scheduling, limits the number of nodes on controller to only 32. A distributed embedded system based on more than 32 nodes needs to implement multiple controller segments interconnected via bridges [7].

### **Virtual Token Passing to Reduce the Token Overhead**

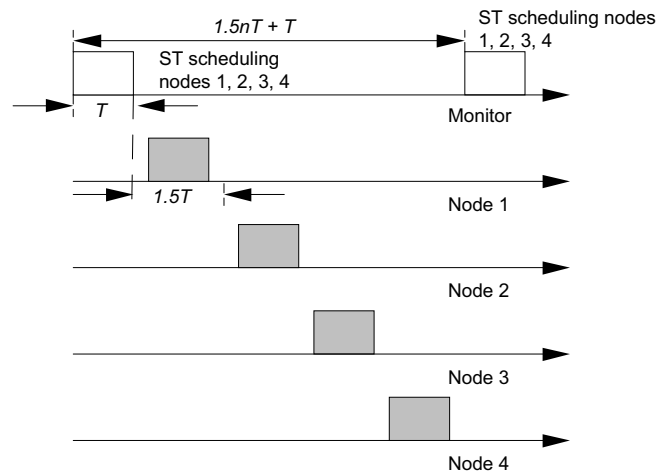
In the following paragraphs suggest solutions to the limitations mentioned previously. These solutions have been devised with the least possible change in the controller frame structure to retain its reasonably high data coding efficiency.

As shown previously, the major overhead in controller is due to explicit token passing related to both MAT and NDT. Instead of explicit token passing, Optimized controller uses a combination of explicit and virtual token passing where the number of virtual tokens in a transmission cycle exceeds that of explicit tokens. Virtual token passing has earlier been used in the P-NET [8][9]. Each master (a node that initiates access to the bus) contains an Access Counter that holds the address of the currently transmitting master. Whenever this counter is incremented (simultaneously on all masters) the master whose address is the same as that of the Access Counter gains access to the bus. Access Counter is incremented upon the following two conditions,

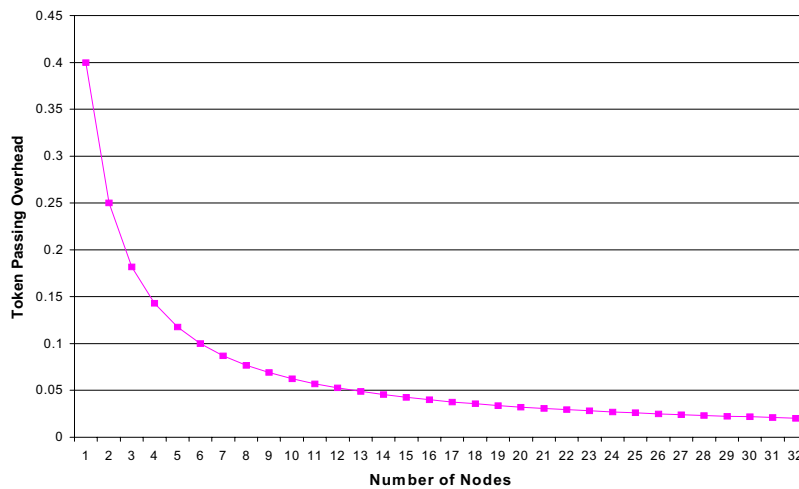
- a) When a request followed by an immediate response from a slave has been completed (i.e., the bus has remained idle for 40 bit periods after the response).

- b) When the master gaining to access the bus. It does not have anything to transmit for 10 bit periods after gaining access to the bus.

Token passing in Optimized controller is slightly different from that of P-NET and is similar to the Node Discovery Procedure in the original controller. A single broadcast token (referred to as the Scheduling Token in Optimized controller) carries the transmission schedule of the nodes on the bus. Each node receives the token and determines (from the order of its address in the payload) the time interval when it shall have access to the bus. Thus, using a single token, up to 32 nodes can be scheduled for transmission with the use of a single ST (Figure 3). However, Optimized CONTROLLER cannot perform the Node Discovery Procedure as the NDT is also a broadcast token. Monitor can detect a node failure by monitoring the communication on the bus and if a node does not transmit any message in a specified number of transmission cycles, it can be labeled as a failed node.



**Figure 3:** Message Scheduling in controller



**Figure 4:** Token Passing Overhead in Optimized controller

No overhead exists due to the Node Discovery Procedure and the overhead related to node scheduling is reduced to one token per transmission cycle. Therefore, for  $n$  nodes scheduled in one transmission cycle, the overhead is given by expression (5) and shown graphically in Figure 4.

$$\text{Overhead of Scheduling Token} = \frac{1}{1.5n+1} \quad (5)$$

It should be noted that  $n$  in (5) does not represent the total number of nodes on the CONTROLLER bus but the number of nodes that are scheduled within a transmission cycle. If  $m$  is used to represent the number of nodes on the bus and  $m < 32$ , then some nodes can be scheduled multiple number of times within a particular transmission cycle. Moreover, if  $m$  is an integer fraction of 32,  $32/m$  schedules can be embedded within the same ST. Therefore the token passing overhead in optimized CONTROLLER can be kept to the minimum especially in case of fixed periodic transmission scheduling where  $n$  is always 32 and  $m$  is an integer fraction of 32.

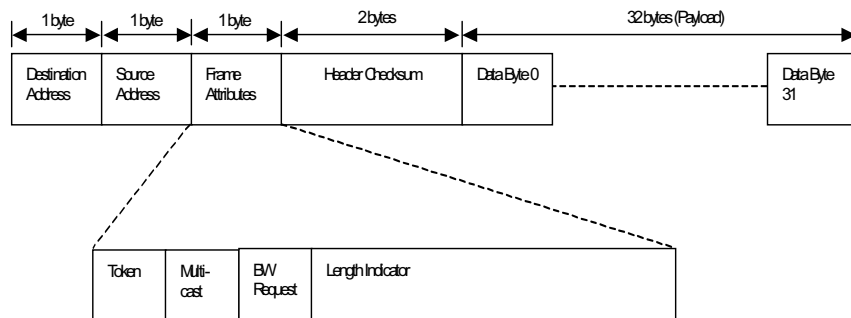
### Dynamic Bandwidth Allocation to Transmit A periodic and Sporadic Messages

Most protocols recently developed for distributed embedded and control systems specify mechanisms to schedule a periodic and sporadic traffic (e.g. [3][4][5]). Controller, in its original form, only schedules nodes periodically for transmission of messages. Furthermore, the protocol makes no distinction between periodic and non-periodic messages. As a result, the protocol is only suitable for providing deterministic access of the medium to nodes. Higher bus utilization with minimum message latency is achieved when applications on nodes generate messages periodically in synchronization with medium access granted to these nodes.

Optimised controller can also schedule the transmission of aperiodic or sporadic messages along with the periodic messages. This is accomplished by generating the transmission schedule for periodic messages based on their periods. Consider  $\vartheta$  being the length of the transmission cycle. If a schedule of nodes containing  $n$  entries is prepared for the transmission of these periodic messages in a particular transmission cycle such that  $1.5nT+T < \vartheta$ , then the time  $\vartheta - 1.5nT - T$  within that transmission cycle can be used to schedule nodes to transmit aperiodic or sporadic messages. This time is referred to as (albeit disjoint) Non-periodic Interval.

As aperiodic and sporadic messages can be generated in bursts, it is possible for a node to require more frequent access to the bus to communicate aperiodic and sporadic messages with the minimum possible queuing delay. Optimised CONTROLLER uses separate queues for periodic, sporadic and aperiodic messages with highest priority to periodic messages and lowest priority to aperiodic messages. In case a node has pending messages, it can request more bandwidth from the monitor.

A previously unused bit in the Frame Attributes field of the header of the original CONTROLLER frame is used to indicate to the monitor that the originating node has backlogged messages and should be scheduled in the Non-periodic Interval of the subsequent transmission cycle. This bit is referred to as the Bandwidth Request bit (Figure – 5).



**Figure 5:** Frame Format for Optimised CONTROLLER

Specific scheduling algorithms that should ensure timeliness of periodic transmissions and minimal average latency to non-periodic transmissions are currently being devised.

### Handling More Than 32 Nodes on a Single CONTROLLER Segment

In CONTROLLER only 32 nodes can be scheduled for transmission. However the number of nodes on the bus can be increased up to 255 if the remaining nodes are not *talkers* but are *listeners*. Listeners only receive information but do not generate any message for transmission. A display is a typical listener. However, more than 32 talkers can also be scheduled on a single bus using multi-cycle scheduling for periodic messages. If  $n$  nodes ( $n < 32$ ) on the bus generate messages with periods  $49mT$  (where  $m \in \mathbb{Z}$ ) and the remaining  $32-n$  nodes generate messages with periods  $49T$ , a total of  $32+n\Delta(m-1)$  nodes can be connected to a single bus and also scheduled feasibly for transmission. Therefore, the monitor needs to generate  $m$  successive schedules arranged in a transmission hyper-cycle consisting of  $m$  transmission cycles each scheduling 32 nodes.

### Conclusions

CONTROLLER was developed as a general purpose networking architecture to provide nodes in distributed embedded and control systems deterministic access to the bus. This paper highlights some of the key deficiencies in the original CONTROLLER. Original CONTROLLER relies on explicit token passing with an elaborate Node Discovery Procedure executed at the end of every transmission cycle to determine the status of nodes that do not respond to tokens during the transmission cycle.

The combined overhead of Medium Access Tokens and the Node Discovery Procedure can be substantial. Optimised CONTROLLER relies on a broadcast token that specifies the transmission schedule to each of the nodes on the bus at the beginning of the transmission cycle. The monitor observes the activity on the bus and if a particular node does not transmit in a specified number of transmission cycles, it is assumed to have failed.

By scheduling nodes for transmission on the basis of the periodicities of the messages they generate, Optimised CONTROLLER can utilize the idle intervals on the bus to transmit non-periodic messages. Furthermore, in case a node has backlogged messages in its queues, it can request more bandwidth from the monitor in the subsequent transmission cycle by setting the Bandwidth Request bit in the CONTROLLER frame header. The monitor schedules extra transmission intervals for such nodes in the Non-Periodic Intervals of the subsequent transmission cycles. Specific scheduling algorithms that guarantee timeliness of periodic messages and minimum average latencies for non-periodic messages are currently being devised and shall be the subject of subsequent publications.

Finally, it has been suggested that more than 32 nodes can be scheduled for transmission on a single CONTROLLER bus if the periods of the messages generated by some of these nodes are greater than the transmission cycle.

## References

- [1] Liu, J.W.S. 2000. "Real-time Systems", Pearson Education.
- [2] Robert Bosch. 1991. "CAN Specification Version 2.0", September, available online from <http://www.can.bosch.com/docu/can2spec.pdf>. accessed 4-3-2007
- [3] Lian, F.; J.R. Moyne; and D. M. Tilbury, 2001. "Performance Evaluation of Control Networks: Ethernet, ControlNet and DeviceNet", IEEE Control Systems Magazine, 66 – 83.
- [4] Leen, G.; and D. Heffernan, 2001. "Time Triggered Controller Area Network", IEE Computing and Control Engineering Journal, 12, no. 6, 245 – 256.
- [5] Fonseca, J.; E. Martins; L. Almeida; P. Pedreiras; and P. Neves, 2000. "Flexible Time-Triggered Protocol for CAN – New Scheduling and Dispatching Solutions", Proceedings of ICC 2000, New Orleans, USA.
- [6] Tanenbaum, A.S. 1996. "Computer Networks", Prentice Hall Inc.
- [7] Younis B.; B. Ahmad; O. Bashir; and F. Azam, 2002. "A Network Protocol for Distributed Embedded Systems", Proceedings of IEEE Students Conference on Emerging Technologies ISCON-2002, Lahore, Pakistan, 149-153.
- [8] Jenkins, C.G., 1997. "P-NET as a European Fieldbus Standard EN50170 vol.1", Institute of Measurement and Control Journal.
- [9] Tovar, E. and F. Vasques , 1998. "A Communication Support for Real-Time Distributed Computer Controlled System", Proceedings of the IEE International Workshop on Discrete Event System (WODES 98), Cagliari, Italy, 178 - 183.