

VIRTUAL TOKEN-PASSING ETHERNET – VTPE

Francisco Borges Carreiro^{1,2}, José Alberto Gouveia Fonseca², Paulo Pedreiras²
{fborges@dee.cefet-ma.br, fborges@ieeta.pt}, jaf@det.ua.pt, pedreiras@det.ua.pt

¹*DEE/ CEFET-MA – Brasil*
Av. Getúlio Vargas N^o 04 Monte Castelo
65025-001 São Luís - MA – Brasil
Phone: +55 98 218 9080 Fax: +55 98 218 9019

²*DET / IEETA – Universidade de Aveiro*
Campus Universitário de Santiago
3810-193 Aveiro Portugal

Abstract: The amount of information that must be exchanged in modern Digital Computer Control Systems (DCCS) and real-time industrial automation is now reaching the limits that are achievable with traditional fieldbuses. Currently, Ethernet is increasingly used in DCCS, however, the available solutions to guarantee real-time behaviour in Ethernet are not well suited to support the DCCS requirements and can't be implemented in low processor power devices. This paper proposes a real-time Ethernet-based protocol (VTPE) that supports real-time requirements, and implies just a small overhead in the system nodes. The proposed protocol is based on the virtual token-passing media arbitration and does not require specialized hardware. *Copyright © 2003 IFAC*

Keywords: Fieldbus, Ethernet, real-time, microprocessors, token-ring protocol.

1. INTRODUCTION

The quantity and functionality of microprocessor-based nodes in modern DCCS have been increasing steadily (Dietrich and Sauter 2000). This evolution has been motivated by new classes of more resource demanding applications, such as multimedia applications (e.g. machine vision), as well as by the trend to use large numbers of simple interconnected

processors, instead of few powerful ones, encapsulating each functionality in one single processor (Pedreiras *et al* 2001). Consequently, the amount of information that must be exchanged among the network nodes has also increased dramatically over the last years and it is now reaching the limits that are achievable using traditional fieldbuses, e.g. CAN, WorldFIP, ProfiBus (Song 2001). Alternative protocols must be found to support this higher bandwidth demand while fulfilling the

timeliness requirements of a real-time communication system. Well-known networks, such as FDDI and ATM, have been extensively analysed for both hard and soft real-time communication systems (Song 2001). However, due to high complexity, high cost, lack of flexibility and interconnection capacity (Song 2001), they have not gained general acceptance for the use at the field level.

Similar efforts have been done with Ethernet (Decotignie 2001), trying to take advantage of the availability of cheap silicon, easy integration with Internet, clear path for future expandability, and compatibility with networks used at higher layers in the factory structure. However, its standardized non-deterministic arbitration mechanism (CSMA / CD) prevents its direct use at field level, at least for hard real-time communications. Despite of this, there are many different approaches for achieving real-time behaviour on Ethernet. However most of the proposed solutions are costly in terms of processing power and memory requirements, and thus they are not well suited for small sensors, actuators and controllers.

There is a need to find Ethernet deterministic solutions, so that it becomes possible to take profit of its with higher data-communication capacity, and use it to replace fieldbuses to interconnect sensors, controllers and actuators at the field level. These solutions must be implemented with simple and efficient mechanisms to access the bus, in order to have reduced code (to fit in most microcontroller's memory) and to impose low communication overhead. The virtual token-passing scheme contains the main characteristics required to this Ethernet implementation.

This paper proposes the Virtual Token-passing Ethernet (VTPE), a new deterministic implementation to support real-time traffic. It uses a virtual-token passing scheme, similar to the one used in P-NET fieldbus (EN50170). However, it uses a producer-consumer cooperation model instead of P-NET's master-slave, in order to reduce the communication overhead. The sensors and actuators can be either directly connected to the microcontroller's ports or via other fieldbus sub network.

The remainder of this paper is organized as follows: Section 2 presents briefly the Ethernet standard, a set of arguments favouring the use of Ethernet at the fieldbus level, and relevant works to achieve real-time behaviour on Ethernet. Section 3 presents the VTPE protocol medium access, the frame format and the real-time parameters. Section 4 presents the VTPE real-time behaviour and a numerical example. Section 5 presents the conclusions and future works.

2. ACHIEVING R-T BEHAVIOUR ON ETHERNET

2.1 Ethernet standard

During the Ethernet evolution two fundamental properties have been kept unchanged: 1) a single collision domain, i.e., frames are broadcast on the physical medium and all the network interface cards (NIC) on it receive the message, and 2) the arbitration mechanism (CSMA/CD).

The use of a single broadcast domain and the CSMA/CD arbitration mechanism have created a bottleneck in highly loaded networks: above a certain threshold, as the load increases the bus throughput decreases (trashing). To overcome this and other disadvantages there are many approaches and techniques to achieve real-time behaviour on Ethernet as it will be discussed in section 2.3.

The standard Ethernet frame is shown in fig.1. It is composed of eight fields: Preamble, Start Frame Delimiter (SFD), Destination Address (DA), Source Address (SA), Type or Length Field, Data, Pad and Frame Check Sequence (FCS).

Alternating 1s/0s	SFD	DA	SA	Type or length	Data	Pad	FCS
Preamble		Frame length (min. 64 bytes e max. 1518 bytes)					

Fig .1 - Ethernet frame format

The destination and source addresses are each one six bytes long. The Type or Length field is two bytes long and specifies the type of protocol or number of data bytes inside of the Data field; the Data field can be between a minimum of 46 bytes and a maximum of 1500 bytes and the FCS is 4 bytes. When the Data field is smaller than 46 bytes it must be filled with zero (Pad) up to complete the minimum permitted Ethernet frame length.

2.2 Why using Ethernet at the fieldbus level

The main obstacle to using Ethernet in real-time communication is that, due to CSMA/CD access protocol, Ethernet cannot provide connected stations with deterministic channel access times and therefore guarantee that data delivery deadlines will be met (Lo Bello and O. Mirabella 2002). In fact, its designer has not envisaged this kind of applications and thus, some properties of this protocol, such as the non-deterministic arbitration mechanism, pose serious challenges concerning its use at this level.

Despite of this, many arguments favouring the use of Ethernet as a fieldbus are commonly defended such as:

- It is cheap, due to mass production;
- Integration with Internet is easy;
- TCP/IP stacks over Ethernet are widely available, allowing the use of application layer protocols such as FTP, HTTP and so on;

- Transmission speed has been steadily increasing in the past and is expected to continue in the future;
- Due to its inherent compatibility with the communication protocols used at higher levels, the information exchange at plant level becomes easier;
- Wide availability of technicians more familiar with this protocol;
- Wide availability of test equipment;
- Mature technology, well specified, equipment available from many sources, no incompatibility issues.

2.3. Achieving real-time on Ethernet

This section presents some efforts to make Ethernet real-time.

Modification of the Medium Access Control

This approach consists on modifying the Ethernet MAC layer to achieve a bounded access time to the bus, e.g. (LeLann, G, and N. Rivierre 1993), (Shimokawa, Y. Shiobara 1985) and (Court, R 1992). For instance, the solution presented in (LeLann, G, and N. Rivierre 1993), CSMA/DCR, consists in a binary tree search of colliding nodes, that is, there is a hierarchy of priorities. Whenever a collision happens the lower priority nodes voluntarily cease contending for the bus, and higher priority nodes try again. This process is repeated until a successful transmission occurs. Two main drawbacks can be identified:

- In some cases the firmware must be modified, therefore the economy of scale obtained when using standard Ethernet hardware is lost;
- The worst-case transmission time, which is the main factor considered when designing real-time systems, can be orders of magnitude greater than the average transmission time. This forces any kind of analysis to be very pessimistic and thus, leads to an under-utilization of the bandwidth.

Adding transmission control over Ethernet

Another way to achieve time-constrained communications over Ethernet consists in adding a layer above it, intended to control the instants of message transmissions, ending up with a bounded number of collisions or even a complete avoidance of them. The major advantage of this kind of approach, when compared to the modification of the MAC layer, is that standard Ethernet hardware can be used. Several different ways of doing transmission control over Ethernet are referred below.

Master/Slave. In this case, all ordinary stations in the system transmit messages only upon receiving an explicit command message issued by one particular station called master. This approach supports

relatively precise timeliness, depending on the master, but introduces a considerable protocol overhead caused by the master messages (notice that the number of messages is duplicated). Moreover, with this approach, the handling of event-triggered traffic is normally inefficient because the master must first become aware of any request before inquiring the respective station.

FTT-Ethernet. This protocol supports both synchronous and asynchronous message exchanges over shared Ethernet (Pedreiras *et al* 2001). Each of these types of traffic is transmitted within specific time windows or phases (designated respectively synchronous and asynchronous). Nodes transmit synchronous messages after explicit request by a special node, designated by Master. This node centrally schedules the synchronous traffic and periodically broadcasts the schedule information using a trigger message (TM). The nodes transmit asynchronous messages autonomously, after application request. To achieve deterministic priority-based asynchronous message transmission, this protocol implements a mini-slotting scheme.

This protocol provides timeliness guarantees to both synchronous and asynchronous traffic. Moreover, the bandwidth used by the relaxed master-slave technique considerably reduces the overhead when compared with the regular master/slave transmission control, since a single TM may trigger the transmission of several data messages. Furthermore, this protocol supports arbitrary scheduling policies and on-line changes to the communication requirements without jeopardizing the timeliness guarantees. However, the complexity resulting from all these features requires resources (memory, CPU) that are not supported by low-end hardware frequently used in distributed embedded systems.

Token-Passing. This method consists on circulating a token among the stations. Only the station currently holding the token is allowed to transmit and the token holding time is upper bounded (IEEE 802.4 timed-token is one example). This scheme is still not very efficient due to the bandwidth used by the token and induces large jitter in the periodic traffic due to variations in the token holding time. Furthermore, token losses generally impose long periods of bus inaccessibility.

TDMA. In this case, stations transmit messages at pre-determined disjoint instants in time in a cyclic fashion. This approach requires precise clock synchronization and does not respond well to dynamic changes in the message set because the communication requirements are distributed and thus, changes must be done globally. On the other hand, it uses the bus bandwidth efficiently since there are no control messages beyond those to achieve clock synchronization.

Switched Ethernet

A solution for the problem of throughput decrease with load network increase, known as thrashing, has

been proposed in the beginning of the 90's, consisting on the use of switches in the place of hubs. A switch creates a single collision domain for each node connected to it. This way, collisions never occur unless they are created on purpose for flow control. Switches also keep track of the addresses of the NICs connected at each port; therefore each NIC only receives the traffic addressed to it. This architecture allows the existence of multiple transmission paths simultaneously, between different network nodes. The global throughput increases significantly since, using switches, the devices on the network no longer share the bandwidth and collisions do not occur. The use of switches became very popular, recently, as a way to improve the performance of shared Ethernet.

When a node transmits a message, this one is received by the switch and then buffered in to the ports where the receivers of the message are connected. If several messages addressed to a given port arrive in a short interval, they are queued and then sequentially transmitted. Concerning the scheduling of messages waiting in an output port, 8 priority queues are available (IEEE 802.1p). The scheduling policy used at this level is a topic currently addressed in the scientific community, e.g. (Jasperneit, J. and P. Neumann 2001).

Unfortunately the use of a switch in an Ethernet network is not enough to make it real-time, in the general case. For instance, output buffers can be exhausted and messages lost if bursts of messages are sent to the same output port. This situation can occur more often than desired. For example, in distributed control systems, the producer/consumers model is typically used. According to this model, one producer of a given datum (e.g. a sensor reading) sends it to several consumers of that datum. This model is efficiently supported in Ethernet by means of special addresses, called multicast addresses. Each network interface card can define a local table with the multicast addresses related to the data that it should receive. However, the switch has no knowledge of these local tables, therefore treats all the multicast traffic as broadcasts, i.e., messages with multicast destination addresses are transmitted to all ports. Therefore, depending on the predominant type of traffic exchanged in a given application (unicast vs. multicast/broadcast), one of the main benefits of using Switched Ethernet, i.e. multiple simultaneous transmission paths, can be seriously compromised.

Other problems concerning the use of switched Ethernet (Decotignie, J. D 2001) are:

- In the absence of collisions the switch introduces an additional latency;
- The number of available priority levels is too small to support the implementation of efficient priority based scheduling;
- The switch only makes Ethernet deterministic under controlled loads.

Others existing proposals

There are many other solutions like the RETHER protocol (Venkatramani, C., T. Chiueh), the Virtual Time Protocol (Malcolm, N. and W. Zhao 1995), (Molle, M. and L. Kleinrock 1985) and the Windows protocols. None of them is well suited for microprocessor implementation because they require fast processors and high memory to implement them, so they are not discussed in this article.

3. VIRTUALTOKEN PASSING ETHERNET-VTPE

In the overview presented in the previous sections the characteristics of the most relevant approaches to achieve real-time communication over Ethernet have been discussed and their drawbacks have been highlighted. It is interesting to notice that such approaches either require specialized hardware, or provide statistical timeliness guarantees, or are bandwidth or response-time inefficient, or are inflexible concerning the properties of the network traffic as well as the traffic scheduling policy, or finally, they are costly in terms of processing power and memory requirements. Thus they are not well suited for use in small sensors, actuators and controllers with communications capability.

Based on these drawbacks and on the demand of distributed embedded systems, the following goals have been established to develop the VTPE.

- Support on the same bus of slow and cheap devices based in microcontrollers, as well as more demanding devices integrating powerful processors;
- Support for on-the-fly operational flexibility like: to add and to remove nodes, to add and to remove messages and change their parameters;
- Indication of temporal accuracy of real-time messages;
- Efficient use of network bandwidth;
- Efficient support of multicast messages;
- Low processing overhead in order to be implemented in microcontrollers with low processing power;
- Support of several data messages inside a single Ethernet frame.

3.1 VTPE medium access control

The VTPE is an Ethernet deterministic proposal based on implicit token rotation (virtual token-passing) like the one used in the P-NET fieldbus protocol. VTPE uses the producer-consumer cooperation model to exchange data over the bus. Producers, in terms of the bus, are active devices that

can access the bus when they are allowed to do it. On the other hand, consumers are passive devices and can only consume the data on the bus¹.

The VTPE system architecture consists of a producer's logical ring like the one depicted in the figure 3. Each node consists of a processor or microcontroller attached to an Ethernet controller.

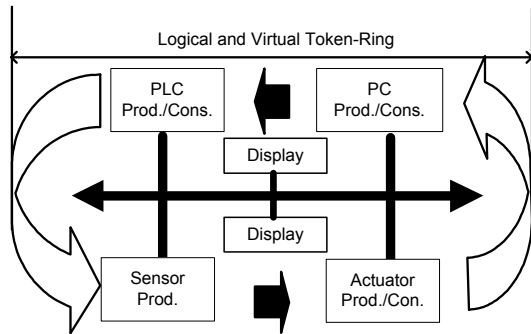


Fig. 3 The Virtual Token-passing in a VTPE system

In the example of figure 3, the distributed system is composed of six nodes, one producer (sensor), three producer/consumer (actuator, PC and PLC) and two consumers (Displays).

To implement a virtual token-passing schema using Ethernet, Ethernet's broadcast destination address must be used because all devices must read each frame dispatched on the bus. In a VTPE system each producer node has a node address (NA), between 1 and the number of producers expected within a system. All producers have an Access Counter (AC), which identifies the node that can access the bus in a specific time interval. Whenever a frame is sent to the bus, an interrupt must be generated in all producer nodes. After the interrupt all nodes increase their ACs and the producer node, whose AC value is equal to its own unique address, is allowed to access the bus. If the actual node doesn't have anything to transmit (or indeed is not present), the bus becomes idle and, after a certain time, all the access counters are increased by one. The next producer is then allowed to access the bus. If, again, it has nothing to transmit, the bus continues idle and the described procedure is repeated until a producer effectively uses the bus.

The procedure described in the previous paragraph accelerates token rotation time when producers have nothing to transmit. However, if it is used just like described, it can lead to a long idle time in the bus. The absence of bus activity can result in clock drift, which, in turn, could lead to AC inconsistencies among system nodes. To prevent this situation the VTPE forces the periodic transmission of a frame with K period to synchronize all access counters. To do this all producers must have an Idle Bus Counter

(IBC), which indicates how many times the bus became idle and no message was sent. All producers also have a *timer*, which can be programmed with time value t_1 or t_2 . t_1 must be long enough to enable the slowest processor in the system to decode the VTPE frame (read the frame). t_2 is used to guarantee the token passing when one or more producers don't have something to transmit. t_1 and t_2 will be discussed further as well as the k value.

To explain the VTPE operation lets see the flow chart depicted in figure 4.

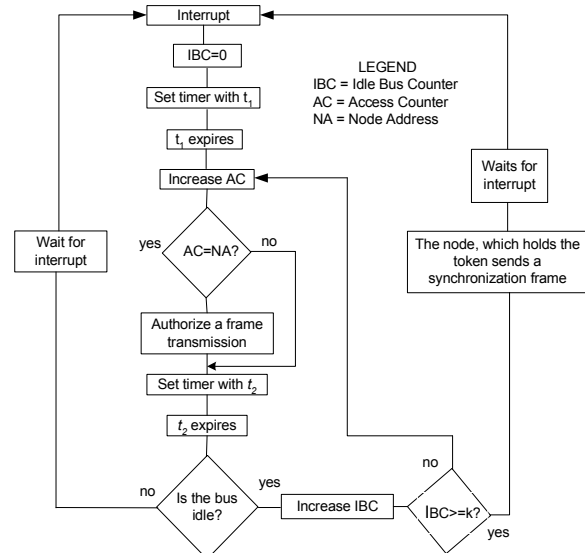


Fig.4 VTPE flowchart

After a frame transmission all producers must reset their IBCs to zero and initialise their *timers* with the t_1 value. After t_1 expires each producer node sets its *timer* with the t_2 value, increases its AC and checks if it is equal to its own node address. Two possibilities can occur:

- The node whose AC is equal to NA must immediately start a frame transmission if it has something to transmit and sets the timer with the t_2 value.
- The nodes with NA different from the current AC value must only set their timers with the t_2 value.

After t_2 expires each producer must check the Bus Status register of the Ethernet controller to verify if there is a frame being transmitted. If true, all producers will wait for the interrupt that will occur. If the bus is idle all producers increase the IBC and compare it with K . If IBC is smaller than K all producers increase the AC and repeat the last procedure until a producer does require access to the bus or the IBC becomes equal or greater than K . However, if $IBC \geq K$, the node that holds the token must send immediately a special frame to synchronize the access counters. The use of the condition $IBC \geq K$ instead of only $IBC = K$ solves the problem of an eventual absence of the node that would be holding the token when $IBC = K$. When the

¹ A device can be simultaneously producer and consumer

access counter exceeds the maximum number of producers, it is preset to 1 and the cycle is repeated again.

Although the VTPE uses the same bus arbitration principle as P-NET (EN 5070170 Volume 1), there are important differences, some due to new features of the protocol and others to the use of Ethernet as transmission medium:

- In VTPE the cooperation model used is the producer-consumer replacing the P-NET master-slave approach;
- In VTPE it is possible to send more than one message in the same packet;
- The VTPE data rate (10 or 100Mbps) is much greater than P-NET data transmission (fixed on 76.8Kbps).
- The VTPE may carry more data per packet (1500 bytes) max than P-NET (63 bytes) max;

3.2 The VTPE format frame

The VTPE protocol uses the MAC Ethernet frame encapsulating a special frame (VTPE frame) inside the Ethernet data field. This is shown in fig. 5.

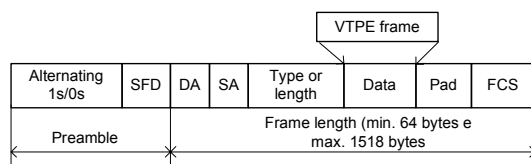


Fig. 5 Virtual Token-Passing Ethernet MAC frame

The VTPE uses the type field instead of length. It represents a reserved constant value, which must be used by all the VTPE messages on the network. The use of this field allows supporting the coexistence, in the same network, of other protocols. On frame reception, the nodes check the type field and only perform further processing if the frame is relevant. Nevertheless, the nodes producing non-VTPE frames must implement the VTPE access control, and transmit frames only if its AC is equal to its NA.

The VTPE frame carries one control field and one or more messages as it is depicted in the figure 6.

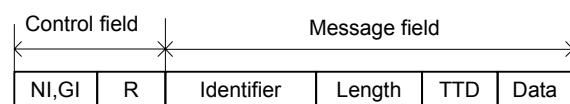


Fig. 6 - VTPE frame format

Since VTPE can send more than one message inside a single Ethernet frame more efficient bandwidth utilization is achieved due to the reduction on padding in case of small messages. Like it is shown

in figure 6, the VTPE frame is composed of two parts: the control field and the messages field.

Control field

The control field is two bytes long and the first byte is divided in two parts. The four less significant bits (NI) identify the number of messages inside the Ethernet frame (up to 16 messages). However this number can be reduced to bind the amount of information that each node must handle on frame reception, favouring the use of small processing power devices. The remaining four bits are the Group Identifier (GI), which will be used to create different producer groups, i.e, sub-networks. The GI idea permits to reduce processing overhead in the nodes by isolating devices that do not belong to the same group. In fact, on frame reception, the nodes check the GI field and only perform further processing if the frame is relevant. Nevertheless, the node must implement the VTPE medium access control.

The second byte of the control field (R) is reserved for future use.

Message field

The message field is composed of the identifier, the length, the TTD (Time To Deadline) and the Data. The identifier is unique and identifies the VTPE message in the system. It is 2 bytes long and thus can address 65536 different messages. The field is two bytes long and is reserved to contain an indication of the time remaining to the message's deadline. The length is two bytes long and indicates the number of bytes in a VTPE message. The VTPE data field is variable, so it can be so small as one byte or so long as 1492 bytes. Observe that the Length field is two-byte long and theoretically it can indicate 65536 bytes. However the maximum data possible per frame in a VTPE message is 1493 bytes (1500 bytes of the maximum data inside a single Ethernet frame minus 7 bytes of the control field and of the VTPE message's header).

To minimize overhead on small processing power devices the messages from and to these nodes must be compatible with their processing capacity. The maximum VTPE message length for these nodes will be fixed further.

3.3 The VTPE parameters t_1 and t_2

To establish these parameters it is necessary to determine the nodes processing workload to run the VTPE protocol, i.e, the workload of communication tasks on frame transmission and reception. This workload is presented next.

On frame reception

The host must execute three basic communication tasks: to attend immediately the Ethernet's controller request, to reset to zero the IBC, to program the *timer* to the value t_1 , and, after t_1 expires, to increment the

AC and to check if it is equal to NA. The remaining activities depend of the protocol type, of the group identifier and of the number of VTPE messages. Table 1 resumes the remaining tasks after a frame reception.

Table 1 Tasks on received frame

Frame type	Tasks
No VTPE	Resets the Ethernet's buffer controller and transmits if it is its chance (AC=NA)
VTPE	The host compares the GI in the frame incoming with the one programmed in its table. If equal, it continues and checks if any messages belong to its message's table. In this case, the messages are transferred to its buffers.

On frame transmission

To reduce the t_1 value the host transfers the VTPE frame to the Ethernet controller before holding the token. Then, when it holds the token, it must just authorize the Ethernet controller to send the frame.

The t_1 parameter is the time required by the host to decode the incoming frame, i.e., to execute the actions shown in table 1. Observe that the time t_1 is processor dependent as well as, indirectly, the number of messages inside the VTPE frame.

The second time, t_2 is the guard time needed to detect nodes absent from the network or that, despite being present, don't have anything to transmit. However, as the Ethernet controller response can differ from one controller to another one, some care must be taken. A higher value of t_2 leads to bandwidth spoiling, and a short value can be difficult to meet in low processing power microcontrollers. A value around 25µs should be adequate for most situations, but this parameter can be adapted according to the particular system characteristics.

4. VTPE REAL-TIME WORST-CASE COMPUTATION

The virtual token-passing idea facilitates to determine the MAC real-time behaviour. To explain the MAC real-time behaviour let's see the figure 7.

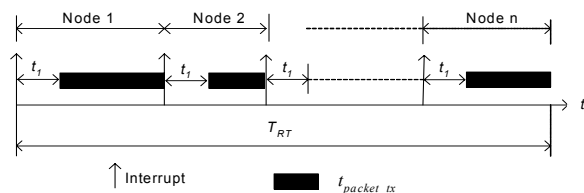


Fig.7 – VTPE real-time behaviour

It is shown in figure 7 that each node transmits a single frame per token holding time, starting at node 1. After node 1 transmission, node 2 gets the right of transmission, and so on up to the last node, the node

n . After node's n transmission, node 1 gets the right of transmission again.

The T_{RT} is the time between two consecutive instants in which a specific node gets the right to transmit. Its value can be found based on the scenario depicted in the fig.7. Thus from the Eq.1 it is possible to calculate the T_{RT}

$$T_{RT} = n * t_1 + \sum_{k=1}^n (t_{packet_tx})_k \quad (Eq.1)$$

Where n is the number of nodes, t_1 is like mentioned before, and $(t_{packet_tx})_k$ is the time to transmit a VTPE packet from node k .

The highest T_{RT} , token rotation time, occurs when all nodes transmit their maximum packet. So the maximum Token Rotation Time $maxT_{RT}$ can be calculated by equation (Eq.2).

$$max T_{RT} = n * t_1 + \sum_{k=1}^n max(t_{packet_tx})_k \quad (Eq.2)$$

The equations (Eq.1) and (Eq.2) shown that the VTPE MAC behaviour is deterministic, besides being very simple to determine the T_{RT} .

4.1 VTPE example

This example shows how to calculate the VTPE parameter and the token rotation time. The system is composed of four processors, two of which are typical 8051 microcontrollers running at 12Mhz; the others are powerful processors. All nodes are supposed to be attached to a 10/100Mbps AX8876L Ethernet controller and the message set is shown in the table 2. In table 2 the node one has four messages and all can be filled and sent in the smallest Ethernet frame to improve throughput performance.

Table 2 VTPE message set

Node	Message Identifier	Consumer	Width (Bytes)	10Mbps (uS)	100Mbps (uS)
(1) 8051	1	(2)	2	57,6	5,76
	2	(2)	2		
	3	(3)	4		
	4	(3)	4		
(2) Processor	5	(4)	250	220,8	22,08
(3) 8051	6	(4)	16	57,6	5,76
(4) Processor	7	(2)	500	400	40

The node 3 has one message only and it can be sent in the smallest Ethernet frame too. In the case of node one and three the time to transmit their messages is 57.6µs and 5.76µs at 10Mbps and 100Mbps respectively. The time to transmit the messages from node 2 and 4 is 220.8µs and 400µs at 10Mbps, 22.08µs and 40µs at 100Mbps.

The messages 3 and 4 from node one are consumed by node three, so it must decode all VTPE frame, i.e. 36 bytes due to VTPE overhead (please see the VTPE format frame).

The messages from node 2 or 4 aren't important to node 1 or 3 (the nodes with microcontrollers) so they are only obligated to decode up to the message identifier.

To run this example in a 8051 it will be necessary 48 move instructions plus 5 compare with jump and plus 5 increment instructions. To execute each move instruction 24 clock cycles are necessary. To execute a compare with jump the processor needs also 24 clock cycles and 12 to the increment instruction. All totalize 1332 clock cycles, which is equivalent to 111 μ S when a 12Mhz crystal is used. The t_1 value is thus 111 μ S and the t_2 is 25 μ S such as suggested before.

The token rotation time can be calculated using the (Eq.2) and its value is:

$$T_{RT} = 1,832mS \text{ at } 10Mbps$$

And

$$T_{RT} = 0,5176mS \text{ at } 100Mbps$$

This example shows that the time impact caused by the 8051 over VTPE is not significant, which implies that it can be implemented in these processors.

5. CONCLUSIONS AND FUTURE WORKS

In this work a new Ethernet deterministic approach was presented. This protocol has been designed for use at field level, resource constrained devices like the ones typically founded in embedded distributed applications.

The most important conclusions over this protocol are:

- The VTPE system architecture can simultaneously rely on low processing power microprocessors or microcontrollers and on powerful processors.
- The introduction of low processing power devices has not a significant impact on the duration of the token rotation time, T_{RT} .
- There is no need for a specific chip-set when implementing the VTPE protocol. This provides the opportunity to use a common standard single chip microprocessor like the 8051 or like a device of the MICROCHIP PIC family.
- It has cheap implementation due to the use of COTS devices produced in large scale;

To continue the work on VTPE the main future developments are:

- Building of a VTPE demonstrator;

- Development of performance analysis;
- To make the VTPE protocol more flexible some additional functions will be included such as remote upload/download and configuration.

REFERENCES

- Court, R.. Real-Time Ethernet. *Computer Communications*, **15** pp. 198-201. April 1992.
- Decotignie, J-D. A perspective on Ethernet as a Fieldbus. *FeT'01, 4th Int. Conf. on Fieldbus Systems and their Applications*. Nancy,France. Nov. 2001.
- Dietrich, D., Sauter, T.. Evolution Potentials for Fieldbus Systems. *WFCS 2000, IEEE Workshop on Factory Communication Systems*. Porto, Portugal, September 2000.
- EN 50170, Volume 1- European Fieldbus Standard
- Jasperneit, J. and P. Neumann. Switched Ethernet for Factory Communication. *ETFA'01, 8th IEEE Conf. on Emerging Tech. on Factory Automation*. Antibes, France. Oct. 2001.
- LeLann, G, N and Rivierre. Real-Time Communications over Broadcast Networks: the CSMA-DCR and the DOD-CSMA-CD Protocols. *INRIA Report RR1863*. 1993.
- Lo Bello, L., O. Mirabella., R.Caponetto. Fuzzy Traffic Smoothing: Another Step towards Statistical Real-Time Communication over Ethernet Networks. *RTLIA - Real-Time LANs in the Internet Age*. Vienna, Austria June 2002.
- Malcolm, N. and W. Zhao. Hard Real-Time Communications in Multiple-Access Networks. *Real Time Systems* **9**, 75-107. Kluwer Academic Publishers. 1995.
- Molle, M. and L. Kleinrock. Virtual Time CSMA: Why two clocks are better than one. *IEEE*
- Pedreiras, P., L. Almeida, and P. Gai, The FTT-Ethernet protocol: Merging flexibility, timeliness and efficiency, *Proceedings of the 14th Euromicro Conference on Real-Time Systems*, Viena, Austria, June 19-21, 2001.
- Song, Y. Time Constrained Communication over Switched Ethernet. *FeT'01, 4th Int. Conf. on Fieldbus Systems and their Applications*. Nancy,France. Nov. 2001.
- Shimokawa, Y. and Y. Shiobara. Real-Time Ethernet for Industrial Applications. *IECON'85*, pp829-834. 1985. *Transactions on Communications*. 33(9):919-933. 1985.
- Venkatramani, C., T. Chiueh. Supporting Real-Time Traffic on Ethernet. *IEEE Real-Time Systems Symposium*. San Juan, Puerto Rico. Dec 1994.