

# The Virtual Token-Passing Ethernet Implementation and Experimental Results

Francisco Carreiro<sup>1,2</sup>

<sup>1</sup>DEE/ CEFET-MA – Brazil

Av. Getúlio Vargas N<sup>o</sup> 04 Monte Castelo

65025-001 São Luís - MA – Brazil

fborges@ieeta.pt

José Alberto Fonseca<sup>2</sup>

<sup>2</sup>DET / IEETA – Universidade de Aveiro

Campus Universitário de Santiago

3810-193 Aveiro Portugal jaf@det.ua.pt

Valter Silva<sup>3</sup>

<sup>3</sup>Escola Superior de Tecnologia e Gestão de Águeda,  
University of Aveiro, Águeda, Portugal

vfs@stga.ua.pt

Francisco Vasques<sup>4</sup>

<sup>4</sup>DEMEGI-FEUP - Universidade do Porto

4200-465 Porto, Portugal

vasques@fe.up.pt

## Abstract

*This paper presents the design and first experimental results of VTPE (Virtual Token-passing Ethernet). VTPE is a deterministic protocol for real-time applications based on shared Ethernet, aimed to be used either in small processing power processors or in powerful ones. It is based on implicit token rotation similar to the virtual token-passing used in the P-NET protocol. The VTPE implementation leads to reduced program code, thus fitting in small microcontrollers' memory and imposing low communication overhead.*

## 1. Introduction

Ethernet is currently the most widely used local area network (LAN) technology in the world. However, it cannot provide a real-time service to the supported applications as the underlying CSMA/CD access protocol does not provide a deterministic arbitration mechanism. Therefore, it cannot guarantee that data delivery deadlines will be met. In fact, such non-deterministic arbitration mechanism poses serious challenges concerning the real-time support of data communications.

Several approaches and techniques have been developed to provide a real-time behaviour to Ethernet-supported applications, taking advantage of the low cost associated to this technology and also of the available data-rates that are much larger than those of most real-time fieldbuses available today. Some of these approaches are based on the modification of the Medium Access Control [1], the addition of transmission control [2], a protocol using time-triggered traffic [3], or the use of switched Ethernet approaches [4].

The objective of this work is to find Ethernet deterministic solutions, so that it becomes possible to use it to interconnect sensors, controllers and actuators at the field level. Such solutions are based on the Virtual Token-Passing Ethernet-VTPE protocol, proposed in a previous paper [5]. The major objective of this paper is to present and discuss the implementation of the VTPE protocol, as well as the preliminary results obtained.

This paper is organized as follows: Section 2 briefly presents the VTPE protocol. Section 3 presents the available implementation of VTPE. Section 4 presents some results obtained in the implementation and Section 5 presents future work and concludes the paper.

## 2. VTPE presentation

The VTPE protocol [5] implements an implicit token rotation procedure (similar to the virtual token-passing used in the P-NET fieldbus protocol [6]) upon the Ethernet MAC layer, guaranteeing the deterministic medium access. In order to preserve the compatibility with existing hardware, the virtual token-passing scheme is implemented using the standard Ethernet's broadcast destination address, guaranteeing that all the devices are able to read each frame dispatched on the bus.

In a VTPE system each producer node is identified by a node address number (NA) between 1 and the number of producers expected within a system and has an Access Counter (AC) that identifies which node can access the bus in a specific time interval. Whenever a frame is sent to the bus, an interrupt will be generated in all producer nodes. After such interrupt, all the producer nodes will increase their AC counters. The 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 it is not present) the bus becomes idle and, after a certain time, all the access counters are automatically increased by one (which corresponds to an implicit token passing). 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. When the access counter exceeds the maximum number of producers, it is reset to 1 and the implicit token-passing cycle is repeated again.

To implement the virtual token passing all producers must 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.

After the end of a frame transmission all the producers will re-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 will immediately start a frame transmission if it has something to transmit and then it sets the timer with the  $t_2$  value.
- The nodes with NA different from the current AC value will set their timers with the  $t_2$  value.

After the expiration of  $t_2$ , each producer will check the Bus Status register of the Ethernet controller to verify if there is a frame being transmitted. If true, all the producers will wait for the interrupt that will occur at the

end of the frame and increase the value of their access counters.

To prevent inactivity in the bus during a long time interval, which could lead to a significant clock drift among system nodes, a timeout is considered too. After the timeout expiration, the node that has the right to access the bus must send a special message.

### 2.1 The VTPE original 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. 1.

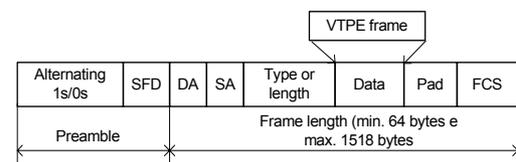


Fig. 1 Virtual Token-Passing Ethernet MAC frame

The VTPE uses the type field instead of the 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. Upon a frame reception, each node will check the type field and will only perform further processing if the frame is relevant. Nevertheless, all the nodes producing non-VTPE frames need to implement the VTPE access control, as frames cannot be allowed to access the bus outside the authorized time intervals, that is, whenever the AC is not equal to the producer NA.

The VTPE frame carries one control field and one or more messages as it is depicted in the figure 2.

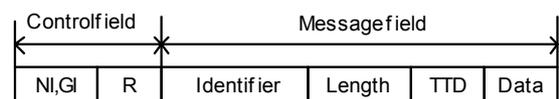


Fig. 2 - VTPE frame format

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 the case of small messages. Like it is shown in figure 2, the VTPE frame is composed of two parts: the control field and the message field.

### Control field

The control field is two bytes long, being the first byte divided in two parts. The four less significant bits (NI) identify the number of messages inside the Ethernet frame (up to 16 messages). 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, upon the frame reception, the nodes will check the GI field and only perform further processing if the frame is relevant. The second byte of the control field (R) is reserved for future use.

### Message field

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

To minimize the overhead on small processing power devices, 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. VTPE implementation

### 3.1 The hardware

The block diagram of a VTPE node is shown in figure 3. A VTPE node consists of an eight-bit microcontroller PIC 18F458 [7] attached to a Packet Whacker Ethernet board [8] based on the RTL8019AS [9] Ethernet controller.

At initialization, the RTL8019AS uses the EEDO, EEDI and EESK lines to find an external EEPROM and gets the MAC address. As the hardware does not use an external EEPROM, the PIC, besides simulating the EEPROM presence, also supplies the MAC address. This

approach simplifies the hardware and makes easier the modification of the MAC address value.

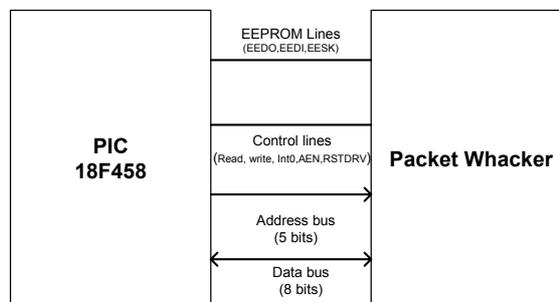


Fig. 3 VTPE node block diagram

The developed VTPE system consists of three nodes interconnected by a hub/switch device, as shown in figure 4.

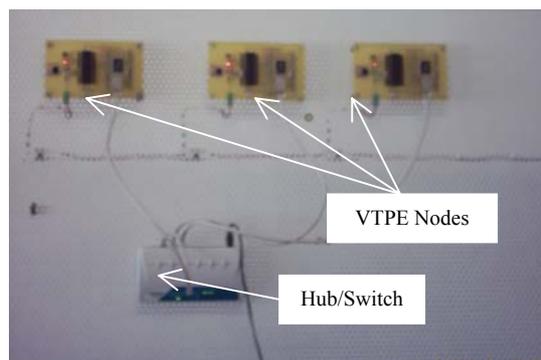


Fig. 4 VTPE system

### 3.2 The software

Currently the VTPE software is still being developed. It is written in C using the PICC18 compiler and developed recurring to the HI-TIDE environment.

Basically the code consists of:

- A function to initialize the Ethernet controller;
- Two functions, one to write data into the Ethernet controller and other to read data from the Ethernet controller, and;
- A function to run the VTPE protocol, *i.e.*, to increase the access counters and to decide if the node has the right to use the bus.

The program code needed to implement the VTPE protocol is very small, approximately 3K bytes, just occupying 9% of the flash memory available in the PIC 18F458.

## 4. Experimental results

The tests carried on until now are aimed to show the system working in the virtual token-passing bus arbitration fashion. For this particular test it was defined that:

- Each node must transmit a predefined VTPE frame per time as depicted in figure 2;
- To facilitate visualization in a sniffer screen, the  $t_i$  time was defined as 1 second.
- The Ethernet type field was defined as 0xabcd.

The Figure 5 shows an Ethereal [10] sniffer screenshot in which some frames transmitted in the VTPE demonstrator were captured. There can be observed that the Ethernet packets were transmitted in the expected sequence of the nodes addresses, i.e., 1:00:00:00:00:00, 2:00:00:00:00:00 and 3:00:00:00:00:00. All frames were sent with broadcast (ff.ff.ff.ff.ff) in the destination address. This is a basic VTPE characteristic because all nodes must accept and check each frame sent in the bus. There can also be noted the Ethernet type field, which for this test, was defined as 0xabcd. In the lower part of the screen it can be seen the transmitted message “BRASIL PORTUGAL”.

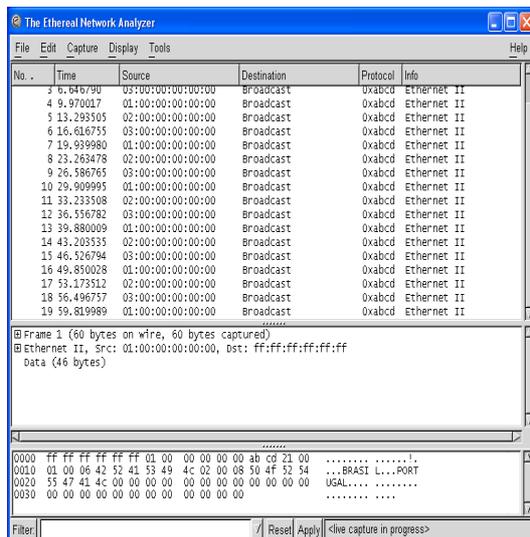


Fig. 5- Output of the Ethernet sniffer showing VTPE operation.

## 5. Conclusion and future work

The VTPE protocol is a proposal to achieve real-time behaviour on top of shared Ethernet. This paper presents the current development state of a demonstration for the VTPE protocol.

The VTPE code for this implementation is very small. It occupies approximately 9% of the available flash memory of the microcontroller used. This is an important result since VTPE is intended to be used in small processing power processors. The demonstration system is also useful to show the operation of VTPE.

To continue the development of VTPE some improvement will be done, for example:

- To implement a mechanism to facilitate the system management;
- To develop and validate mechanisms to promote fault tolerance in real conditions of operation, namely a solution based on the publication of the individual access counter values.

## References

- [1] Jae-Young Lee, Hon-ju Moon, Sang Young Moon, Wook Hyung Kwon, Sung Woo Lee, and Ik Soo Park. “Token-Passing bus access method on the IEEE 802.3 physical layer for distributed control networks”. Distributed Computer Control Systems 1998 (DCCS’98), Proceedings volume from the 5<sup>th</sup> IFAC Workshop. Elsevier Science, Kidlington, UK, pp. 31-36, 1999.
- [2] Venkatramani, C., T. Chiueh. Supporting Real-Time Traffic on Ethernet. *IEEE Real-Time Systems Symposium*. San Juan, Puerto Rico, December 1994.
- [3] 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
- [4] Choi Baek-Young, Song Sejun, N. Birch, and Huang Jim. “Probabilistic approach to switched Ethernet for real-time control applications”. *Proceedings of Seventh International Conference on Real-time Computing Systems and Applications*, pp. 384-388, 2000.
- [5] Carreiro, F. Borges, Fonseca, J. Alberto, and Pedreiras, P, “Virtual Token-Passing Ethernet-VTPE”, FET 2003 5<sup>th</sup> IFAC International Conference on Fieldbus Systems and their Applications, Aveiro, Portugal, July 2003.
- [6] EN50170, Volume 1- European Fieldbus Standard
- [7] Microchip, “PIC 18Fxx8 Data Sheet”, 2003.
- [8] Fred Eady, “Introducing the Packet Whacker Part1 and Part 2” Circuit Cellar online, October 2001
- [9] Realtek, “RTL8019AS Specification”, 2001.
- [10] The Ethereal packet Sniffer, <http://www.ethereal.com>