

VIRTUAL-TOKEN PASSING ETHERNET - VTPE

Francisco Carreiro¹

José Fonseca²

Paulo Pedreiras²

¹Centro Federal de Educação Tecnológica do Maranhão -BRASIL

²Universidade de Aveiro - PORTUGAL

THE PROBLEM

- Current trend to use microcontrollers and processors of low power processing interconnected with powerful ones;
- More resource demanding applications, such as multimedia (e.g. machine vision);
- The amount of information that must be exchanged among the network nodes has also increased and it is now reaching the limits of traditional fieldbuses, e.g. CAN, WorldFIP, ProfiBus

THE PROBLEM

- Networks such as FDDI and ATM have not gained general acceptance for the use at the field level due:
 - high complexity
 - high cost
 - lack of flexibility
- Similar efforts have been done with Ethernet, but there is still a lack of efficient solutions for real-time applications using small sensors, actuators and controllers.

OUR SOLUTION

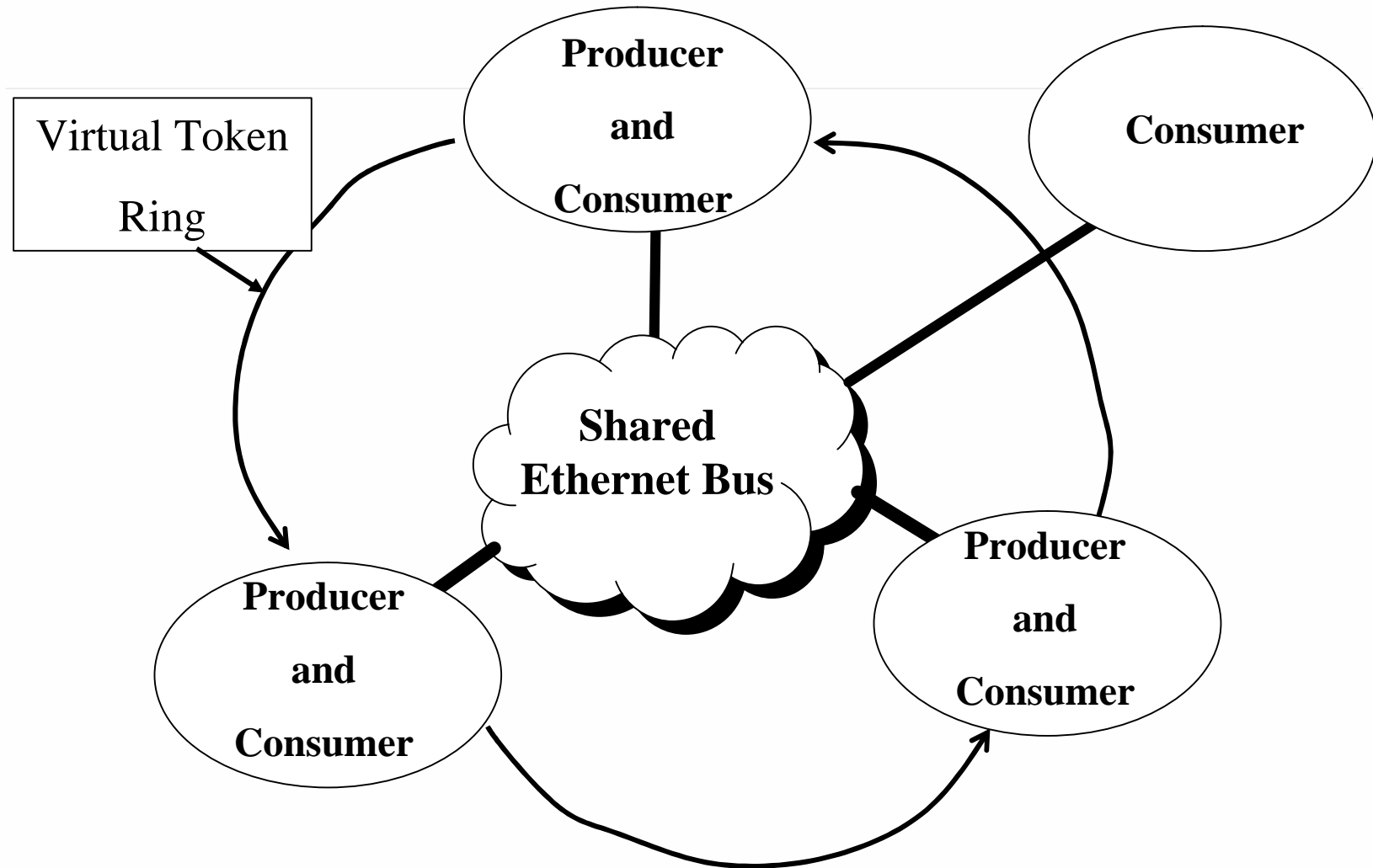
We propose an Ethernet deterministic solution based on virtual token-passing like the one used in the P-NET fieldbus protocol.

→ Simple system architecture allows simultaneous use of low processing power microcontrollers together with powerful ones

OUR SOLUTION

- VTPE also supports
 - efficient use of network bandwidth,
 - efficient multicast messages;
 - several data messages inside a single Ethernet frame (piggybacking)
 - indication of temporal accuracy of real-time messages;

VTPE SYSTEM ARCHITECTURE



VTPE FEATURES

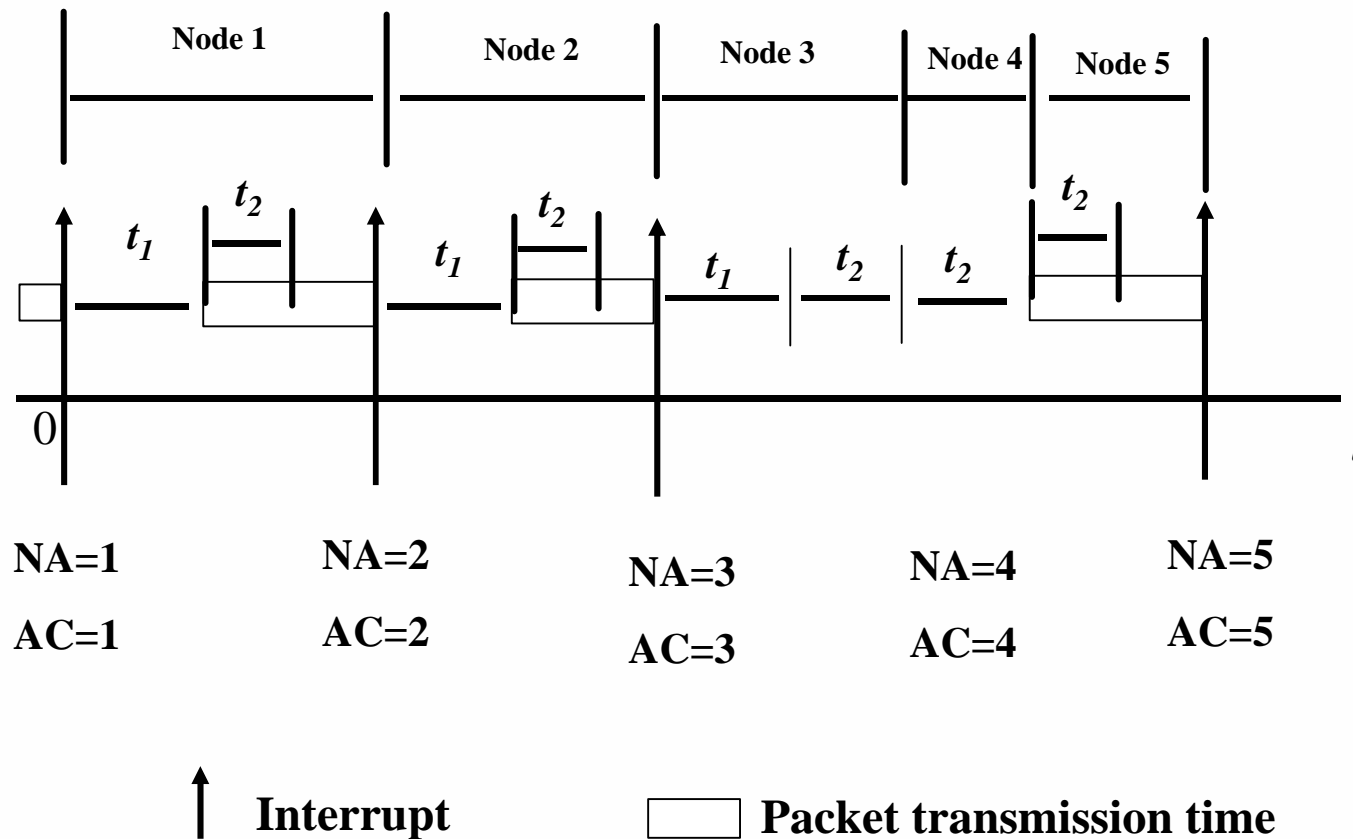
Some VTPE features.

- Each VTPE node can send only one frame per token visit. This frame is encapsulated on Ethernet packet and all nodes must receive it.
- Each producer has a node address (NA) and an Access Counter (AC)
- After a frame transmission an interrupt must be generated in all nodes, after this:
 - each producer node increase its access counter
 - The node which $AC = NA$ can use the bus.

VTPE FEATURES

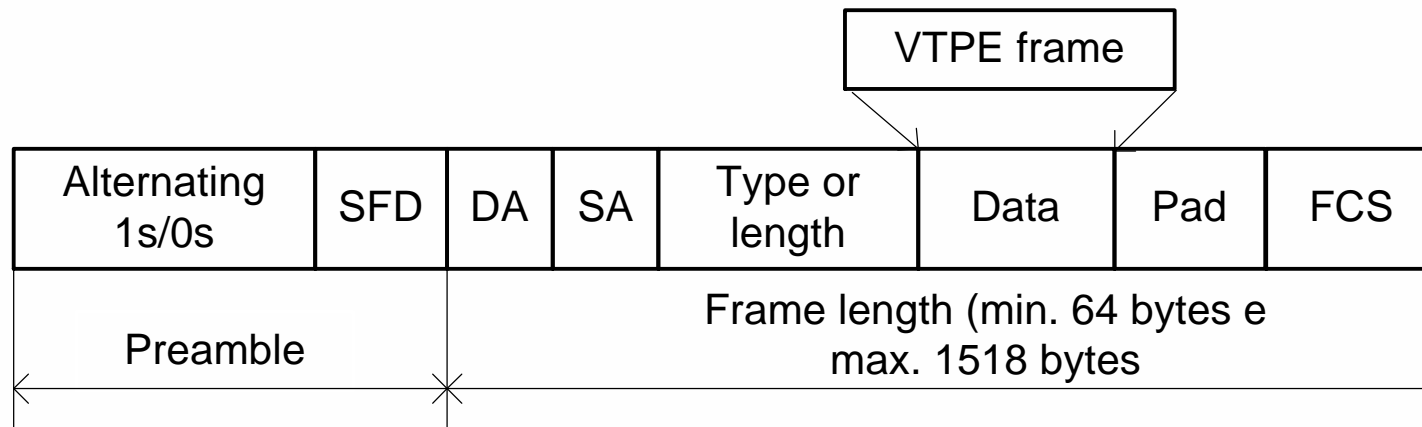
- Each node also has two timers t_1 and t_2 .
- To avoid long idle bus period, which can cause AC inconsistency, each producer also has an Idle Bus Count (IBC)
- Periodically will be sent a synchronization frame when $IBC \geq k$ where k is an integer deduced from max clock drifts,

VTPE REAL-TIME BEHAVIOUR

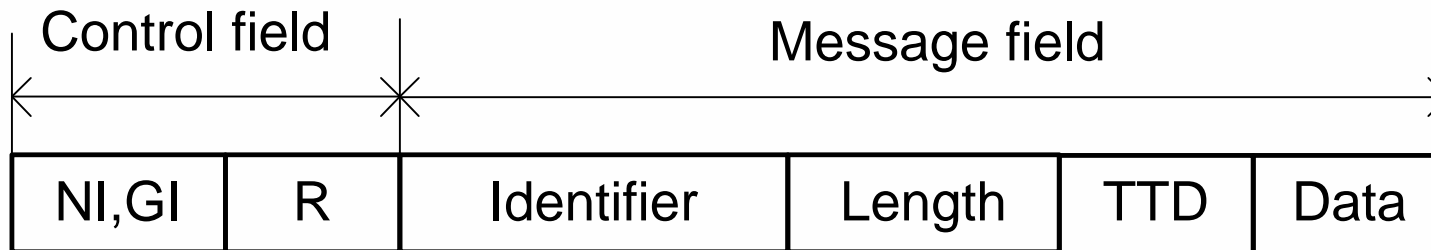


VTPE FORMAT FRAME

VTPE frame inside Ethernet data field



VTPE FORMAT FRAME



NI – Number of messages inside in VTPE frame – 4 bits

GI – Group identifier – 4 bits

R – Reserved – 1 byte

Identifier – Message identifier

Length – 2 bytes

TTD – Time To Deadline – 2 bytes

Data – Application data

VTPE ANALYSIS

Token rotation time equation

- General case

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

- Worst-case

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

T_{RT} WORST-CASE CALCULATION

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

T_{RT} WORST-CASE CALCULATION

Computing t_1

To run this example in a typical 8051 microcontroller with 12Mhz clock will be necessary at least 48 move instructions, 5 compare with jump and 5 increment instructions.

In terms of time all is equivalent to $111\mu\text{S}$. So the the minimum t_1 value is $111\mu\text{S}$.

T_{RT} WORST-CASE CALCULATION

Using the equation (2) the T_{RT} worst-case results:

At 10 Mbps is: $T_{RT} = 1,832mS$

At 100 Mbps is: $T_{RT} = 0,5176mS$

CONCLUSIONS

This work presents a protocol designed for use at field level in resource constrained devices like the ones typically found in embedded distributed applications. We believe that it fills these requirements due to its small processing overhead, low memory requirements and low cost.

FUTURE WORKS

Main future works

- Building of a VTPE demonstrator;
- Development of performance analysis;
- To turn VTPE more flexible some additional functions will be included, such as, remote upload/download and configuration, mechanism to attribute more bandwidth to a node.