Final Version
RTSS'99, WIP Session, Madrid, December 99

Evaluating P-NET Message's Response Time with Fixed Priority Queuing at Application Process Level

Eduardo Tovar
Department of Computer Science,
ISEP, Polytechnic Institute of Porto
Rua São Tomé, 4200 Porto, Portugal
E-mail: emt@dei.isep.ipp.pt

Francisco Vasques
Department of Mechanical Engineering,
FEUP, University of Porto
Rua Bragas, 4099 Porto Codex, Portugal
E-mail: vasques@fe.up.pt

Alan Burns
Department of Computer Science,
University of York
Heslington, York, YO1 5DD, UK
E-mail: burns@cs.york.ac.uk

Abstract: *P-NET is a multi-master fieldbus standard based on a virtual token passing scheme. In P-NET each master is allowed to transmit only one message per token visit. In the worst-case, the communication response time can be derived considering that, in each token cycle, all stations use the token to transmit a message. In this paper, we define a more sophisticated P-NET model, which considers the actual token utilisation. We then analyse the possibility of implementing a local priority-based scheduling policy to improve the real-time behaviour of P-NET.*

1. Introduction

P-NET [1] fieldbus networks adopt a multi-master MAC scheme. All communication is based on a principle, where a master sends a request and the addressed slave immediately returns a response. For multi-master support, P-NET uses a Virtual Token Passing (VTP) scheme. Contrarily to other network protocols, such as Token Passing Bus (IEEE802.4) or Fibber Distributed Data Interface (FDDI), in P-NET there is no explicit token transmission between masters.

In this paper we address the possibility of enhancing the real-time capabilities of P-NET by adding local priority-based scheduling mechanisms which may reduce the negative impact of the First-Come-First-Served (FCFS) implementation of P-NET's outgoing queue.

This paper is organised as follows. In the next section we describe the basic concepts of the P-NET medium access control (MAC). In section 3 we introduce a basic timing analysis, which is updated with more recent results [2] in section 4. Finally, in section 5 we introduce the undergoing work, which consists on adding local priority-based scheduling mechanisms at the application process level.

2. Basic Concepts of the P-NET MAC

In P-NET, the Virtual Token Passing (VTP) scheme is implemented using two protocol counters. The first one, the Access Counter (AC), holds the node address of the currently transmitting master. When a request has been completed and the bus has been idle for 40 bit periods (*bp*) (40 *bp* = 520μs @ 76,8Kbps[1]), each one of the ACs is incremented by one. The master whose AC value equals its own unique node address is said to hold the token, and is allowed to access the bus.

[1] The P-NET standard uses a data rate of 76800 bps. This data rate resulted from weighing up the conflicting requirement for data to be transported as fast as possible, but not at such speed as to negate the use of standard microprocessor UARTS, or restrict the usable distance or cable type [3].

The second counter, the Idle Bus Bit Period Counter (IBBPC), increments for each inactive bus bit period. Should any transactions occur, the counter is re-set to zero.

If a master have nothing to transmit (or indeed is not even present), the bus will continue inactive. Following a further period of 130μs (10 bit periods), the IBBPC will have reached 50, (60, 70,…) all the ACs will again be incremented, allowing the next master access. The virtual token passing will continue every 130μs, until a master does require access.

The P-NET standard also stands that each master is only allowed to perform one message transaction (later on defined as message cycle) per token "visit". After "receiving" the token the master must transmit a request before a certain time has elapsed (the standard specifies that the master's worst-case reaction time should be at most 7 bit periods).

A slave is allowed to access the bus, between 11 and 30 bit periods after receiving a request, measured from the beginning of the stop bit in the last byte of the frame. The maximum allowed delay is then 390μs (corresponding to 30 bit periods). Later on, this delay will also be denoted as the slave's turnaround time.

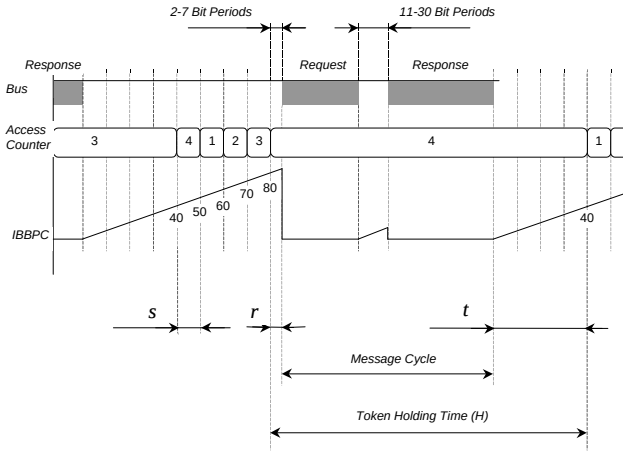

Figure 1

A message cycle is composed by a master's request followed by the addressed slave's response. Assume that C_M is the maximum transmission duration of all message cycles in a P-NET network. This duration includes both the longest request and response transmission times, and also the worst-case slave's turnaround time.

If a master uses the token to perform a message cycle, we can define a token holding time[2] as:

$$H = r + C_M + t \quad (1)$$

In equation (1), t (= 40 *bp*) corresponds to the time to pass the token after a message cycle has been performed. r (≤ 7 *bp*) denotes the worst-case master's reaction time.

If a station do not use the token to perform a message cycle, the bus will be idle during s (= 10 *bp*) before all ACs are incremented.

For better understanding both the basic MAC procedures and the notation used, refer to figure 1.

3. Basic Timing Analysis

We assume the following message model:

$$S_i^k = (C_i^k, T_i^k, D_i^k) \quad (2)$$

S_i^k defines a message stream i in master k (k = 1 .. n). A message stream is a temporal sequence of message cycles concerning, for instance, the remote reading of a specific process variable. C_i^k is the longest message cycle duration of stream S_i^k. T_i^k is the periodicity of stream S_i^k requests. Finally, D_i^k is the relative deadline of the message cycle, that is, the maximum admissible time span between the instant when the message request is placed in the outgoing queue and the complete reception of the related response at the master's incoming queue. ns^k is the number of message streams associated with a master k.

We also consider that messages generated in the distributed system can be periodic or sporadic. For the case of sporadic message requests, its period corresponds to the minimum time between two consecutive requests.

In our model the relative deadline of a message can be equal or smaller than its period ($D_i^k \leq T_i^k$). Thus, if in the outgoing queue there are two message requests from the same message stream, this means that a deadline was missed. It also results that the maximum number of pending requests in the outgoing queue will be, in the worst-case, ns^k.

We denote the worst-case response time of a message stream i in a master k as R_i^k. This time is measured starting at the instant when the request is placed in the outgoing queue until the instant when the complete response appears in the incoming queue. Basically, this time span is made up of the two following components: the time spent by the request in the outgoing queue, until gaining access to the bus; the time needed to process the message cycle (send the request and receive the related response).

Assume a network scenario with 3 masters. If in master 1 the number of message streams is 2 ($ns^1 = 2$), the worst-case response time results from considering that two requests are placed in the outgoing queue just after receiving the response of a previous message cycle. The last message in the outgoing queue will need to wait for 2 token visits to be processed. Figure 2 depicts the worst-case response time analysis, assuming all token holding times equal to H.

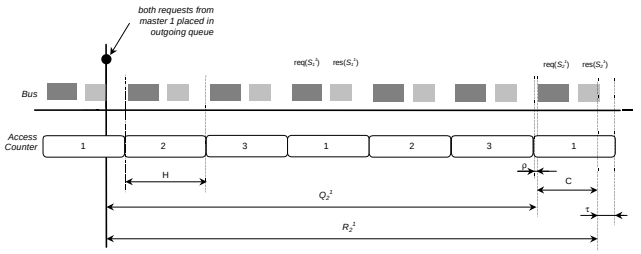

Figure 2

In figure 2, Q_i^k is the worst-case queuing delay for a message stream i in master k. In fact, the worst-case queuing delay (and the worst-case response time R_i^k) is the same for all streams in a particular master. In each station the worst-case queuing delay only depends on ns^k. Thus, we define the worst-case queuing delay of a message cycle in master k as:

$$Q^k = ns^k \times V - H + t + r = ns^k \times V - C \quad (3)$$

where V is the worst-case virtual token rotation time, which is defined as follows:

$$V = n \times H \quad (4)$$

and n stands for the number of masters of the network. The worst-case response time (WCRT)[3] is then given by:

$$R^k = Q^k + C = ns^k \times V \quad (5)$$

Therefore, in order to guarantee the message streams deadlines, the following condition must be satisfied:

$$R^k \leq D_i^k, \ \forall_{k,i} \quad (6)$$

4. Considering Actual Token Utilisation

In the previous section we assumed that in ns^k consecutive token cycles, all other masters use the token for processing message cycles and therefore this situation corresponds to the worst-case token cycle rotation. Such analysis is only accurate for a specific master k if the following condition is verified:

$$ns^y \geq ns^k, \forall_{y \neq k} \quad (7)$$

Since in all the other masters the worst-case number of pending requests may be greater than ns^k, all those masters may use the token in the next ns^k consecutive rotations of the token. As messages are processed in a first-come first-served basis, in master k the worst-case response time is then given by equation (5). Otherwise, if $ns^y < ns^k$ (y ≠ k), such analysis may not be accurate because, the worst-case response time depends also on the periodicities of the stream requests in those masters.

The basic timing analysis (section 3), which considers that the token is always fully utilised, is similar to a TDMA analysis, with the length of a TDMA slot equal to H. Considering this TDMA analogy, we define the worst-case queuing delay of a message in a master k, as:

$$Q_{TDMA}^k = ns^k \times V - H + t + r = \left(ns^k \times n - 1\right) \times H + t + r \quad (8)$$

To develop a more accurate analysis for the case where one or more masters do not use the token, we need to

[2] It is not usual to include the token passing time in the token holding time. However, due to the specificity of the Virtual Token Passing scheme, we decided to associate the token holding time with the state of the P-NET access counter in each node.

[3] As the bit rate in P-NET is 76800 bps, the propagation delay can be neglected, even for P-NET networks with some kilometers.

evaluate how many times this happens during ns^k consecutive token cycles. Furthermore, considering that for each non used token, the token holding time ($H = r + C_M + t$) is reduced to s, the actual worst-case queuing delay will be:

$$Q^k = Q^k_{TDMA} - nut^k \times (H - s) \quad (9)$$

where nut^k (non used tokens) stands for the number of times that during ns^k consecutive token cycles other stations rather than k do not use the token. In [2] the authors derived the exact computation for *nut*:

$$Q^k = Q^k_{TDMA} - \sum_{y=1}^{n} \left[\max\left\{0, ns^k - nrq^{y/k}\right\} \right] \times (H - s) \quad (10)$$

where

$$nrq^{y/k} = ns^y + \sum_{i=1}^{ns^y} \left\lfloor \frac{J^{y/k}_{ring} + Q^k}{T^y_i} \right\rfloor \quad (11)$$

Figure 3, explains how we got to equation (12). Basically, what this figure reflects is that the critical instant we should consider, is when in all other masters y ($y \neq k$) ns^y requests are placed in the outgoing queues $J^{y/k}_{req} = [(n+k-y) \bmod n] \times H$ before instant 0. We denote this time as the *ring request jitter*. To this time we must subtract $J^{y/k}_{vis} = H$, the *ring visit jitter*, because a new request in a master y, will only be able to be processed if it appears H before Q^k. A full treatment for these jitters is given in [2]. Thus,

$$J^{y/k}_{ring} = [((n+k-y) \bmod n) - 1] \times H \quad (12)$$

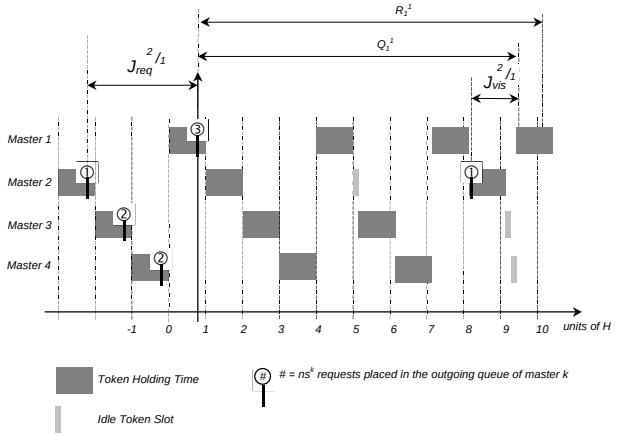

Figure 3

5. Undergoing Work

One possibility to reduce the worst-case response time relies on the implementation of a priority queue at the application process level. The P-NET protocol uses First-Come-First-Served (FCFS) outgoing queues at the communication stack. Thus, we need to implement a priority-based queue for the message requests at the application process level, and limit the stack outgoing communication queue to one pending request.

One of the most used priority assignment schemes is to give the messages a priority level based on its period: the shorter the period, the higher the priority (that is, $T_i < T_j \Rightarrow P_i > P_j$). This type of priority assignment is known as *rate monotonic* (RM) priority assignment. If, due to the characteristics of the devices, some of the inputs are sporadic, where the period is considered to be equal to the minimum inter-arrival time, it may not be reasonable to consider $D = T$.

A different priority assignment can then be to give the tasks a priority level based on its deadline: the shorter the deadline, the higher the priority (that is, $D_i < D_j \Rightarrow P_i > P_j$). This type of priority assignment is known as *deadline monotonic* (DM) priority assignment.

In order to use a priority-based dispatching policy, there is the need to previously check the feasibility of the priority assignment. There are several results available for the task scheduling in a single processor environment that can be adapted for message scheduling in communication networks.

5.1. Single Processor Pre-Run-Time Scheduling Theory

Joseph and Pandaya [4] showed that the worst-case response time r_i of a task t_i is found in a scenario in which all tasks are at their maximum rate and released synchronously at instant $t = 0$. r_i is computed by the following recursive equation (where *hp(i)* denotes the set of tasks with higher priority than task t_i):

$$r_i^{m+1} = C_i + \sum_{j \in hp(i)} \left(\left\lceil \frac{r_i^m}{T_j} \right\rceil \times C_j \right) \quad (13)$$

The recursion ends when $r_i^{m+1} = r_i^m = r_i$ and can be solved by successive iterations starting from $r_i^0 = C_i$. Indeed, it is easy to show that r_i^m is non-decreasing. Consequently, the series either converges or exceeds D_i. In this last case, the task t_i is not schedulable.

This result is valid for the pre-emptive context. However, the pre-emptive context is not much useful for the case of message cycles instead of tasks. In fact a message cycle transmission can not be interrupted if a higher priority message appears in the outgoing queue. Hence, we need an analysis for the non pre-emptive context.

Contrarily to the pre-emptive context, fewer results are known about fixed priorities non pre-emptive scheduling. In [5] the authors updated the analysis of Joseph and Pandaya [4] to include blocking factors introduced by periods of non pre-emption. The following equations represent this analysis:

$$r_i = w_i + C_i \quad (14)$$

where w_i is given by:

$$w_i^{m+1} = B_i + \sum_{\forall j \in hp(i)} \left(\left\lceil \frac{w_i^m}{T_j} \right\rceil \times C_j \right) \quad (15)$$

B_i is the blocking factor of task t_i (a bound on the time a lower priority task can execute and prevent the execution of task t_i). In [6] a methodology for deriving B_i is given. Although the inclusion of this blocking factor was to solve the problem of the tasks being not totally independent, we can use this result for the scheduling analysis of non

pre-emptive messages. In the case of the tasks scheduling, the blocking factor will be:

$$B_i = \max_{\forall j \in lp(i)} \{C_j\} \quad (16)$$

where *lp(i)* denotes the tasks with lower priority than *i*.

5.2. Fixed Priority Scheduling in the P-NET Model

Equation (10) must now be updated to:

$$Q_i^k = B_i^k + \sum_{\forall j \in hp(i)} \left(\left\lceil \frac{Q_i^k}{T_j^k} \right\rceil \times V \right) - \sum_{y=1}^{n} \left[\max\left\{0, ns^k - nrq^{y_{/k}}\right\} \right] \times (H - s) \quad (17)$$

where the blocking factor is *V* (as defined in (4)) or 0 for the lower priority message (in this case there is no blocking - priority inversion).

5.3. Holistic Approach

We can assume that in a master, message requests are placed in the priority-ordered application process queue by an application task. It is, therefore important to have a holistic approach [7,8] for the evaluation of the end-to-end communication delay. Underlying the holistic approach, in [9] it is defined the end-to-end communication delay, which includes the following four major components: *generation delay*; *queuing delay*; *transmission delay*; *delivery delay*:

The generation delay is the worst-case time taken between the "notification" of the sender task and the queuing of the related message. The queuing delay can be seen as the time that the message spends waiting to be removed from the queue by the communications device. The transmission delay is the time taken for the message to be sent once it has been removed from the outgoing queue. Finally, the delivery delay is the amount of time it takes to process the incoming data and deliver it to destination tasks.

This model can be adapted to encompass the P-Net model. In P-NET, a slave differs from a master in the sense that it does not have to support pending packets, as responses to requests from masters should be immediate. In fact, this generic computational model is particularly suitable for P-NET masters (slaves are passive devices).

The worst-case end-to-end communication delay (*E*) can be expressed as follows:

$$E = g + Q + C + d \quad (18)$$

In equation (18), *g* represents the worst-case generation delay for the master application task to generate and queue a specific message request. The term *Q* corresponds to the worst-case delay for that request to gain access to the communications device after being queued. The term *C* corresponds not only to the worst-case for transmitting a request, but also to the time needed to receive the response from the slave. Finally, *d* represents the delivery delay, that is, the time needed to process the response before finally delivering it to the destination task, which, in the case of P-NET, is in the same host processor as the sending task.

In [7] the holistic approach allows the analysis of end-to-end computations in real-time distributed systems where host processors schedule tasks and messages according to fixed priorities by means of a very interesting concept: *attribute inheritance*. The message sent by a task inherits two of its temporal attributes, namely the period and the release jitter. If each instance of the task communicates, the message inherits a period equal to that of the task. Furthermore, if the message can be queued at any time by the sender task, the difference from its earliest and latest releases is bounded by the sender worst-case response time (assume a pre-emptive context for the tasks at the host processor). This is the release jitter inherited by the message. Basically, in terms of the network subsystem, what may happen is that the minimum inter-arrival time for a message may be shorter than the period of the sending tasks. A release jitter should then be included in equation (17):

$$Q_i^k = B_i^k + \sum_{\forall j \in hp(i)} \left(\left\lceil \frac{Q_i^k + J_j^k}{T_j^k} \right\rceil \times V \right) - \sum_{y=1}^{n} \left[\max\left\{0, ns^k - nrq^{y_{/k}}\right\} \right] \times (H - s) \quad (19)$$

and also in equation (11):

$$nrq^{y_{/k}} = ns^y = \sum_{l=1}^{ns^y} \left\lceil \frac{J_{ring}^{y_{/k}} + Q_i^k + J_l^y}{T_i^y} \right\rceil \quad (20)$$

In this way, the overall analysis can be de-coupled in two different subsystems: the host processor and the network.

Previous works on this subject, considered network systems where the sending and destination tasks where at different processors. Furthermore, they do not consider actual token utilisation (in case of token passing networks [7, 9]) or the networks considered had priority busses [10], hence with no need to implement the priority-based queue at the application process level.

The work being now carried out consists on obtaining an accurate characterisation of the release jitter in P-NET networks, as results by the implementation of the scheduling mechanisms in actual P-NET node masters.

References

[1] EN 50170. General Purpose Field Communication System. Vol. 1/3 (P-NET). CENELEC 1996.

[2] E. Tovar, F. Vasques and A. Burns. “Communication Response Time in P-NET Networks: Worst-Case Analysis Considering Actual Token Utilisation”. Technical Report ISEP - 981001, 1998.

[3] C. Jenkins. P-NET as a European Fieldbus Standard EN 50170 vol. 1. In: Institute of Measurement + Control Journal, 1997.

[4] M. Joseph, P. Pandaya. “Finding Response Times in a Real-Time System”, The Computer Journal, Vol. 29, NO. 5, pp. 390-395, 1986.

[5] N. Audsley, A. Burns, M. Richardson, K. Tindell, A. Wellings. “Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling”, Software Engineering Journal, Vol. 8, NO. 5, pp. 285-292, September 1993.

[6] L. Sha, R. Rajkumar, J. Lehoczky: “Priority Inheritance Protocols: an Approach to Real-Time Synchronisation”, IEEE Transactions on Computers, Vol. 39, NO. 9, pp. 1175-1185, September 1990.

[7] K. Tindell, J. Clark. “Holistic Schedulability Analysis for Distributed Hard Real-Time Systems”, in Microprocessors and Microprogramming, No. 40, 1994.

[8] M. Spuri. “Holistic Analysis for Deadline Scheduled Real-Time Distributed Systems”, INRIA, Technical Report no. 2873, April 1996.

[9] K. Tindell, A. Burns, A. Wellings. “Analysis of Hard Real-Time Communications”, in The Journal of Real-Time Systems, No. 9, 1995.

[10] K. Tindell, H. Hansson, A. Wellings. “Analysing Real-Time Communications: Controller Area Network (CAN)”, in Proceedings of the IEEE Real-Time Systems Symposium, pp. 259-263, December 1994.