End-To-End Response Times in Real-Time Distributed Systems

Jean-François Hermant and Marco Spuri*
Projet REFLECS - *INRIA* - B.P. 105 - 78153 Le Chesnay Cedex - France

Abstract

Distributed real-time applications are usually related to the control of some external environment. Since they are normally composed of tasks statically allocated on several host processors and communicating via message passing, the timing requirements are defined in terms of end-to-end guarantees. That is, end-to-end computations must be carried out within specified maximum delays in order to not jeopardize the system interactions with the external environment.

Such forms of guarantee require a predictable architecture both in the processing and in the networking domains. The Timed Token MAC protocol, adopted in the FDDI local area network standard, is among those protocols providing such predictability. Thus, it represents a suitable choice for real-time systems.

In this paper we propose an innovative analysis for establishing end-to-end guarantees in deadline scheduled real-time distributed systems in which network accesses are granted to host processors by using the Timed Token MAC protocol. A detailed description of how to exactly bound messages worst-case response times is also given.

1 Introduction

In hard real-time distributed systems, response times must be bounded in order to guarantee *a-priori* that the strict timing constraints specified by the system designer are met. In such systems, the timing requirements are expressed in terms of maximum response times of end-to-end computations [2]. End-to-end computations are usually composed of periodic and sporadic tasks which communicate by message passing. The total delay of such computations, termed *end-to-end response time*, includes several components, among which we find the time taken by a sending task to queue a message, the time taken by the message to gain access to the communications device and to be transmitted, and the time taken by the receiving task to process the message at the receiving end [11].

Tindell *et al.* [11] and Tindell and Clark [10] described a very interesting approach, termed *holistic analysis*, to find such bounds in real-time distributed systems in which tasks are statically allocated on host processors and scheduled highest priority first, with priorities fixed at design time. Their analysis has been described for three different network access protocols: a timed token protocol, a real-time fixed priority protocol, and a TDMA protocol.

In this paper we adopt a similar approach, more deeply described in [9], for the analysis of real-time distributed systems in which the local schedulers on host processors adopt the Earliest Deadline First algorithm [6], and network accesses are granted by using the Timed Token medium access control protocol [4]. We also assume that at each network adapter the local queue of outgoing packets is ordered earliest deadline first.

The motivations of our choices are as follows. End-to-end deadline guarantees are possible only if the communication network supports the timely delivery of inter-task messages [1]. Although not ideal, owing to its predictability the Timed Token MAC protocol has been addressed by several authors as a suitable choice for real-time systems. The optimality of the EDF scheduling algorithm is known also when complex task models are assumed [8]. For these reasons we assume local EDF task and message schedulers.

The key aspect of the analysis we are going to describe is the computation of worst-case response times both for application tasks and for inter-task messages. The problem has been solved for deadline scheduled uniprocessor systems in [8].

In this paper we propose a system model in which application tasks are statically allocated to host processors and locally scheduled by using the EDF algorithm. Inter-task messages are locally queued earliest deadline first at each host network adapter as well. An analysis ables to compute tight bounds on end-to-end response times is then described. The paper is structured as follows. In Section 2 our assumptions are stated and a first outline of the analysis is given. In Section 3 we present the details of how to compute the worst-case inter-task message response times. The computation of overall end-to-end response times is described in Section 4. Finally, In Section 5 we state our final remarks.

2 Assumptions and Problem Decomposition

In our analysis we assume a distributed system configuration in which several host processors are connected by a physical or logical ring (in this second case the actual network is a shared broadcast bus). Access to the ring is arbitrated by using the Timed Token medium access control protocol [4].

The Timed Token protocol normally provides two classes of service: the *synchronous* class and the *asynchronous* class [5]. The former class is intended for messages with regular arrivals and delivery time constraints. The latter class is intended for messages with arbitrary arrival laws and without time constraints. In this paper we assume that messages involved in end-to-end critical computations are handled by the synchronous service.

*This author has been supported by the Commission of the European Communities under contract ERBCHBGCT930458, proposal ERB4050PL931117.

A special bit pattern called *token* circulates around the ring, providing access control among the hosts. At network initialization time, all hosts negotiate a common value for the *target token rotation time* ($TTRT$). Each host p is then assigned a fraction H_p of $TTRT$, termed *synchronous bandwidth*, which is the maximum time the station can transmit synchronous messages upon reception of the token. After the transmission of synchronous messages, if any, the station can send asynchronous messages only if it has received the token earlier than $TTRT$ units of time after the last token visit. The duration of the possible asynchronous message transmission is limited to $TTRT$ minus the time elapsed between the previous and the current token visits.

In its simplest form an end-to-end computation is composed of a sending task, a message, and a destination task[1]. We assume the system designer specifies a maximum delay for the whole computation, relative to the arrival time of the sending task. The message and the destination task inherit the sending task's period.

Application tasks are assumed to be statically allocated to host processors. A task i consists of an infinite number of *requests*, or *instances*, whose arrival times are separated by a minimum time T_i, termed *period* (according to the conventional notation, this assumption is common to periodic and sporadic tasks). Each instance of i must be completed within a deadline D_i relative to its arrival time.

Similarly, messages exchanged by application tasks are characterized by period, number of packets (we assume messages are broken into packets of equal fixed size), relative deadline, and release jitter. The arrival of a message occurrence may not necessarily coincide with its *release time*, that is, the actual instant of time at which the occurrence is queued. The difference between the latest and the earliest relative queuing times is termed *release jitter*.

Host processors are assumed to be scheduled according to the Earliest Deadline First algorithm [6], which at any time schedules the task instance with the earliest absolute deadline. The EDF algorithm is known to be optimal even in presence of release jitter and relative deadlines not related with the corresponding periods [8]. Owing to this optimality, we also assume that the outgoing packet queues at each local network adapter are similarly ordered earliest deadline first. Note that the deadline scheduling of network messages has been recently suggested [3, 14] in order to achieve a higher degree of network schedulability.

In order to establish the feasibility of the system we will show how to compute upper bounds on the worst-case end-to-end response times. These bounds are then simply checked against the required maximum end-to-end delays.

Finding tight bounds on worst-case end-to-end response times is not a trivial task. A very interesting approach has been proposed by Tindell *et al.* [11]. The components of the end-to-end computation are first identified:

- *Generation delay* - the time needed by the sender task to generate and queue the message.
- *Queuing delay* - the time needed by the message to gain access to the network.
- *Transmission delay* - the time needed for the transmission of the message.
- *Delivery delay* - the time needed by the destination processor to deliver the message to the destination task.
- *Processing delay* - the time needed by the destination task to process the message, that is, to complete its execution.

Host processors and network subsystems are then analysed separately in order to compute worst-case response times of tasks and messages. The different analysis are linked by the concept of *attribute inheritance*: the message sent by a task inherits a release jitter equal to the worst-case response time of the task, and similarly a destination task inherits its release jitter from the message it receives. An increase in response times may cause an increase in jitters, which may cause another increase in response times, and so on. The analysis of the different subsystems are thus repeated several times, until stability is reached by the system parameters. At this point the end-to-end response times can be easily found according to the decomposition previously described.

The approach just mentioned is termed *holistic analysis*, and is more deeply described in [10] for fixed priority systems, and in [9] for deadline scheduled systems. We believe the major merit of the holistic approach is to make the process of analysing a distributed system as a whole tractable, since the overall complexity is pseudo-polynomial, without being at the same time too pessimistic in the computation of worst-case response time bounds.

The essential tool needed by such analysis is the computation of worst-case response times of tasks and messages. The details of how to carry out the computation for deadline scheduled host processors can be found in [8]. In the following section we describe instead how to compute message worst-case response times when the Timed Token MAC protocol described earlier is utilized to grant network accesses. Finally, in Section 4 we give the details of how to bound the end-to-end delays.

3 Communications Analysis

According to the attribute inheritance, a message m inherits from its sender task $s(m)$ period and release jitter. The period is $T_m = n_m T_{s(m)}$, if m is sent once every n_m periods of $s(m)$. The release jitter J_m is equal to the difference between the latest and the earliest possible queuing times of m, which without any further hypothesis is equal to the worst-case response time $r_{s(m)}$ of the sender task.

The worst-case response time of a message m can be computed with an approach similar to that used for the application tasks. In particular, we need to examine all scenarios like that depicted in Figure 1 (arrivals are denoted by upward arrows, while deadlines are denoted by downward arrows), in which an occurrence of m arrives at time a and has deadline $d = a + D_m$. The transmission of this occurrence is most possibly delayed when the following conditions are both verified:

[1] More generally, an end-to-end computation can be composed of a "chain" of tasks communicating by message passing.

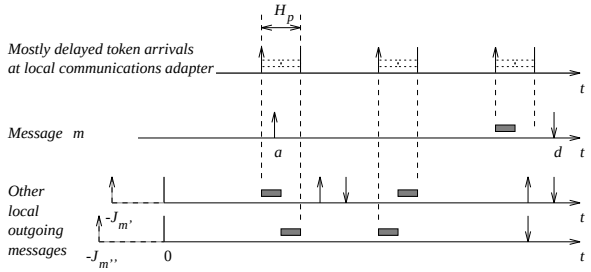

Figure 1: Worst-case scenario for a message m's occurrence.

- The token visits the host processor of $s(m)$ as late as possible.
- The other local outgoing messages cause the highest possible interference on m's packets, that is, they are released as soon as possible and at their maximum rate.

Let p be the host processor where $s(m)$ is allocated. We can account for the two conditions described above by introducing the following functions: $\bar{I}_p(t)$, representing the interference of the other host processors and of the local asynchronous traffic on the synchronous network accesses of host processor p in an interval of time t, and $\overline{HW}_m(a,t)$, the higher priority workload locally preceding the transmission of the m's packets arrived at time a.

$\bar{I}_p(t)$ can be correctly defined by using the result of Zhang and Burns [12], according to whom the largest time interval between any v consecutive token arrivals at host processor p is:

$$t_{l+v-1,p} - t_{l,p} = (v-1)\,TTRT + \sum_{q \neq p} H_q + \tau - \left\lfloor \frac{v-1}{n+1} \right\rfloor \epsilon,$$

where $t_{l,p}$ is the time the token makes its lth arrival at host p, τ is the portion of $TTRT$ unavailable for transmitting messages, and $\epsilon = TTRT - \sum_{q=1}^{n} H_q - \tau$. Hence, the visits of the token at host p, useful for synchronous transmissions, are mostly delayed when:

- the first visit occurs at time $t = 0^-$, that is, just before any synchronous packet has been released, thus making useless this first visit, and
- the asynchronous capacity is fully utilized at any host, whenever available.

See Figure 2 for a graphical representation. The visits useful for synchronous message transmissions are $t_{1,p}, t_{2,p}, \ldots$ Accordingly, if v is such that $t_{v-1,p} + H_p \leq t < t_{v,p} + H_p$, the interference of the other processors and the local asynchronous traffic in the interval $[0,t]$ is then

$$\bar{I}_p(t) = t_{v,p} - (v-1)H_p,$$

with $t_{0,p} = 0$.

As regard the definition of $\overline{HW}_m(a,t)$, we just need to observe that in the interval $[0,t]$ a message m' is queued $\alpha_{m'}(t) = 1 + \lfloor (t + J_{m'})/T_{m'} \rfloor$ times, but at most $\beta_{m'}(a + D_m) = 1 + \lfloor (a + D_m + J_{m'} - D_{m'})/T_{m'} \rfloor$ occurrences have deadlines less than or equal[2] to $a + D_m$, thus

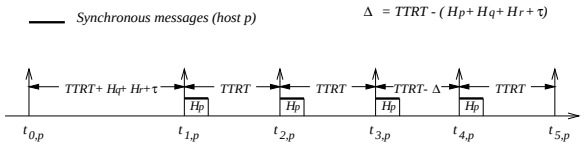

Figure 2: Mostly delayed token visits at host processor p.

$$\overline{HW}_m(a,t) = \rho \sum_{\substack{m' \neq m \\ m' \in Out(p) \\ D_{m'} \leq a + D_m + J_{m'}}} \min\{\alpha_{m'}(t), \beta_{m'}(a + D_m)\}\, C_{m'},$$

where $Out(p)$ is the set of outgoing messages of host processor p, and ρ is the transmission time of a single packet.

Since the transmission of any packet is non-preemptable, in order to effectively evaluate the communication delay of the m's occurrence arrived at time a, $r_m(a)$, we follow the approach described in [11]: we separate the queuing and the transmission delays of m by exactly bounding the communication delay of the first $k-1$ packets and the transmission delay of the kth packet. The idea is to exactly bound the worst-case time needed by the kth packet to gain access to the network and then to include the time needed for its transmission to the destination end. In practice, the communication delay of the first k packets of m is thus

$$r_{m,k}(a) = \max\{J_m + B_m + k\rho + \mathcal{P}, L_{m,k}(a) + \rho + \mathcal{P} - a\},$$

where $L_{m,k}(a)$ is the length of the higher priority network busy period which precedes the transmission of the kth packet of m, that is, the maximum time needed by the kth packet to gain transmission access to the network, while $\rho + \mathcal{P}$ is the time for its actual transmission and for its propagation through the network to its destination ($\mathcal{P}$ is the network propagation delay). B_m is the blocking time of m: due to the non-preemptive character of packets transmission, m may be delayed by the transmission of a lower priority packet. Hence, $B_m = \rho$.

The maximum among the two quantities is necessary because $J_m + B_m + k\rho + \mathcal{P}$ is an obvious lower bound to the value of $r_{m,k}(a)$. The worst-case communication delay of the whole message m can be easily computed by substituting $k = C_m$, that is, $r_m(a) = r_{m,C_m}(a)$, where C_m is the number of packets message m is broken into.

The higher priority network busy period preceding the transmission of m's kth packet include the two interferences we have previously described, and the previous packets of message m. In the scenario we are examining, the kth packet is preceded by $\gamma_{m,k}(a) = \lfloor (a + J_m)/T_m \rfloor\, C_m + k - 1$ other

[2] We do not assume any particular policy for breaking deadline ties. Hence, in the worst-case packets sharing the same deadline should be considered as having higher priorities.

packets of the same message. The equation for the computation of $L_{m,k}(a)$ is thus

$$L_{m,k}(a) = \overline{HW}_m\left(a, L_{m,k}(a)\right) + \gamma_{m,k}(a)\rho + \bar{I}_p\left(L_{m,k}(a)\right), \quad (1)$$

which can be solved by successive iterations, starting from $L^{(0)}_{m,k}(a) = 0$.

Note that owing to the non-preemptability of packet transmissions, as explained the transmission time of the kth packet has not been included in Equation (1). However, this implies that the interval in which the two functions $\bar{I}_p(t)$ and $\overline{HW}_m(a,t)$ are evaluated must be closed, since also interferences possibly starting at time t will precede the transmission of the kth packet.

The last issue to be addressed in this description is the determination of the significant values of the variable a for which we have to compute $r_m(a)$. Since in the computation of $r_{m,k}(a)$ we evaluate the length of a busy period, $L_{m,k}(a)$, we obtain an upper bound on the significant values of a by founding the maximum length of any possible adapter busy period, L_p.

The length L_p can be determined by taking into account the local message workload and the remote interference, and by using the well known fixed point computation:

$$\begin{cases} L_p^{(0)} &= \rho \sum_{m \in Out(p)} C_m, \\ L_p^{(m+1)} &= W_p\left(L_p^{(m)}\right) + I_p\left(L_p^{(m)}\right), \end{cases}$$

where $W_p(t)$ and $I_p(t)$, contrary to $\overline{HW}_m(a,t)$ and $\bar{I}_p(t)$, respectively, are defined on the interval $[0, t[$, that is, they only include occurrences arrived before t:

$$W_p(t) = \rho \sum_{m \in Out(p)} \left\lceil \frac{t + J_m}{T_m} \right\rceil C_m$$

and

$$I_p(t) = t_{v,p} - (v-1)H_p,$$

where v is such that

$$t_{v-1,p} + H_p < t \leq t_{v,p} + H_p.$$

The significant values of a are then found in the interval $[-J_m, L_p - J_m - B_m - \rho C_m[$. That is, the worst-case response time of message m is

$$r_m = \max\left\{r_m(a) : -J_m \leq a < L_p - J_m - B_m - \rho C_m\right\}.$$

In order to further reduce the number of evaluations of $r_m(a)$, we can observe that the local maxima of the higher priority busy period length $L_{m,k}(a)$ are found for those values of a for which there is at least another local message with one occurrence having the deadline at time $a + D_m$, or for which also message m has an occurrence queued, *i.e.* released, at time $t = 0$, that is, all messages are first released synchronously. Hence, the significant values of a are those in the interval $[-J_m, L_p - J_m - B_m - \rho C_m[$ for which $\exists m'$, $\exists k$ such that $a = -J_{m'} + kT_{m'} + D_{m'} - D_m$. These are in fact all and the only points at which the right side of Equation (1) has discontinuities in a.

4 End-To-End Response Times

In Section 2 we have identified in any end-to-end computation of two communicating tasks five components: generation delay, queuing delay, transmission delay, delivery delay, and processing delay. In Section 3 we have established that in absence of further hypothesis the generation delay is upper bounded by the worst-case response time of the sender task. The queuing delay and the transmission delay are included in the worst-case response time of the message. Finally, the delivery delay and the processing delay are included in the worst-case response time of the destination task.

According to the attribute inheritance, the destination task $d(m)$ inherits the period of the message m it receives and a jitter equal to the difference between the latest and the earliest message response times, that is, $J_{d(m)} = r_m - (\rho C_m + \mathcal{P})$. As previously addressed, its worst-case response time can be computed as described in [8].

This computation explicitly includes the time needed to process the message. However, we also have to take into account the delivery delay. We can assume that each incoming packet in a host processor is handled by a suitable interrupt handler, which implements the upper layers of the communications protocols, and which actually delivers the message occurrence to the destination task after the last packet has been correctly received. By including the overhead of packet handling in the analysis of destination hosts, the delivery delay is implicitly taken into account. Details can be found in [9].

It is worth noting once again that there is a strong dependency between host processor and network scheduling. The attribute inheritance is such that the message communication delays strongly depend on the sender task response times and, vice versa, the worst-case response times of destination tasks strongly depend on message communication times. This dependency, already addressed in Section 2, is the base of the holistic analysis [10, 9]. In order to coherently analyse the system as a whole we must proceed in iterative steps. At each step we examine all subsystems using the methodologies described in [8] for host processors and in this paper for the network. The timing attributes are those computed at the previous step. The computation is halted either when all values stabilize or when a subsystem is found unschedulable.

Once the holistic procedure is halted, the last step is to check the worst-case end-to-end computation delays against their required maximum values. In particular, the five components of an end-to-end computation delay can be finally summed up as in the following formula:

$$r_{\text{end-to-end}} = r_{s(m)} + (r_m - J_m) + \left(r_{d(m)} - J_{d(m)}\right). \quad (2)$$

The formula is also graphically illustrated in Figure 3. The worst-case response time of the sender task, $r_{s(m)}$, is the generation delay. $(r_m - J_m)$ includes the queuing and the transmission delays. Finally, the delivery delay (implicitly) and the processing delays are included in the last term $\left(r_{d(m)} - J_{d(m)}\right)$.

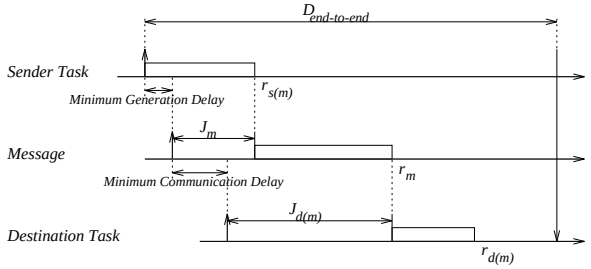

Figure 3: Components of the end-to-end computation delay.

5 Conclusions

In this paper we have discussed a novel approach for the analysis of real-time distributed systems in which application tasks are locally deadline scheduled in host processors, and outgoing messages are also locally queued earliest deadline first. Network accesses have been assumed to be arbitrated by the Timed Token MAC protocol, adopted by a number of network standards, including FDDI. The main characteristic of the approach is to compute tight bounds on end-to-end response times by means of accurate analysis of host processor and network subsystems. A procedure for the exact computation of worst-case message response times has also been described.

Preliminary experiments have shown that the choice of deadline scheduling for host processors and outgoing messages may be effective in practice. In particular, the optimality of the EDF algorithm should let higher resources utilization in comparison with other approaches like fixed priorities.

Few problems remain opened. The Timed Token protocol may be a suitable choice for real-time distributed systems, owing to its predictability. However, the communication delays are strongly affected by the values of some parameters, namely the target token rotation time and the hosts synchronous bandwidths, which must be carefully set. A number of works describing several bandwidth allocation schemes have already appeared [7, 13, 1, 12]. Unfortunately, all of them assume a single outgoing synchronous stream in each host processor. Hence, the question of how to properly set the protocol parameters with earliest deadline local outgoing queuing remains opened.

Another aspect which deserves further investigations is the deadline assignment. As described, end-to-end requirements are specified in terms of end-to-end deadlines. It is not obvious how to assign deadlines to intermediate tasks or messages, with the double objective of achieving feasibility without resources utilization wastes, often caused by too restrictive constraints.

A final remark is about tasks allocation. The objective asserted above is also affected by the way application tasks are assigned to computational resources. A number of heuristics can be found in the literature. We believe that the two problems of allocation and deadline assignment are strongly connected. Hence, heuristics for joint solution searches are needed.

References

[1] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing Synchronous Message Deadlines with the Timed Token Medium Access Control Protocol," *IEEE Trans. on Computers* 43(3), March 1994.

[2] M. Di Natale and J.A. Stankovic, "Dynamic End-to-end Guarantees in Distributed Real-Time Systems," *Proc. of the IEEE Real-Time Systems Symposium*, 1994.

[3] D. Ferrari, "A New Admission Control Method for Real-Time Communication in an Internetwork," in S. Son, Ed., *Advances in Real-Time Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1995.

[4] R.M. Grow, "A Timed Token Protocol for Local Area Networks," *Proc. Electro/82*, May 1982.

[5] R. Jain, *FDDI Handbook: High-Speed Networking Using Fiber and Other Media*, Addison-Wesley, Reading, Massachusetts, 1994.

[6] C.L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of ACM* 20(1), pp. 40-61, January 1973.

[7] N. Malcolm and W. Zhao, "Guaranteeing Synchronous Messages with Arbitrary Deadline Constraints in an FDDI Network," *Proc. of the IEEE Conf. on Local Computer Networks*, 1993.

[8] M. Spuri, "Analysis of Deadline Scheduled Real-Time Systems," Rapport de recherche 2772, INRIA Rocquencourt, Le Chesnay Cedex, France, January 1996, submitted to *IEEE Trans. on Software Engineering*.

[9] M. Spuri, "Holistic Analysis for Deadline Scheduled Real-Time Distributed Systems," Rapport de recherche, INRIA Rocquencourt, Le Chesnay Cedex, France, April 1996.

[10] K. Tindell and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems," *Microprocessors and Microprogramming* 40, 1994.

[11] K. Tindell, A. Burns, and A.J. Wellings, "Analysis of Hard Real-Time Communications," *The Journal of Real-Time Systems* 9, 1995.

[12] S. Zhang and A. Burns, "An Optimal Synchronous Bandwidth Allocation Scheme for Guaranteeing Synchronous Message Deadlines with the Timed-Token MAC Protocol," *IEEE ACM Trans. on Networking* 3(6), December 1995.

[13] Q. Zheng and K.G. Shin, "Synchronous Bandwidth Allocation in FDDI Networks," *Proc. of ACM Multimedia 93*, 1993.

[14] Q. Zheng and K.G. Shin, "On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks," *IEEE Trans. on Communications* 42(2/3/4), 1994.