On Handling Real-Time Communications in MAC Protocols

Sonia Mammeri and Rachid Beghdad
Faculty of Sciences, University of Bejaia, Algerie

Abstract: *In this paper, we present a critical study of two real time Medium Access Control (MAC) protocols: the reservation protocol of IEEE802.5, and the timed token protocol. First of all, the reservation protocol of IEEE802.5 suffers from the lower priority messages that may experience starvation, which is caused by large amounts of enqueued higher-priority packets. To solve that, we propose a protocol that allows the transmission of low priority messages after a high priority message. Secondly, our improved timed token protocol suffers from the high number of enqueued non real-time messages. Facing this problem, we propose a new version that allows the transmission of non real-time messages without waiting the next token arrival. Simulation results show the robustness of our proposed solutions.*

Keywords: *Real-time constraints, reservation protocol, timed-token protocol, scheduling messages, MAC protocols.*

Received April 17, 2010; accepted January 3, 2011

1. Introduction

Unlike some known Local Area Networks (LANs), the real-time LANs differ in the real-time constraints required by the transmission of messages. Indeed, in such networks, it is imperative to guarantee a limited time for the transmission of messages described as urgent or having priority. Ensuring a bounded transmission time requires an appropriate scheduling of real-time messages. Scheduling means organizing in time the realization of tasks, given time constraints (deadlines, sequencing constraints) and constraints on the availability of required resources. Trivially, the "deterministic" Medium Access Control (MAC) protocols are those which should be able to meet these requirements, as it is the case with the Timed Token Protocol (TTP) [2], and Profibus [1]. However, some "indeterministic" protocols were also adapted for these real-time requirements as it is the case with CSMA/CD which led to the development of CSMA with Deterministic Collision Resolution (CSMA/DCR) protocol. These MAC protocols are known to govern the access conflicts and to ensure an upper bound response time in the worst case.

The major real-time networks/protocols can be classified into three possible classes. Class 1 is related to fieldbuses like Profibus [1, 3, 10], FIP [4, 5], and CAN [9]. These fieldbuses provide real-time capabilities in wired networks. Class 2 concerns token networks like IEEE802.5, TTP [2], and FDDI [1]. The protocols implemented in these networks perform a scheduling based on the notion of guaranteeing a bounded access time to the MAC stations. The third class concerns multiple access networks/protocols like CSMA/DCR, ATM, and Rether [13].

We propose in this paper a critical study of two real-time MAC protocols: the reservation protocol of IEEE802.5, and the TTP (once again). Firstly, we found that lower priority packets may experience starvation in case of the reservation protocol of IEEE802.5, and proposed a better approach. Secondly, we proved mathematically in [2] that our protocol is better than TTP, but without any simulation. So, we will present here some simulations to prove our claim. In addition, we criticize our protocol proposed in [2], especially concerning the time allocated to the transmission of non real-time messages, and propose another improved TTP. Our simulations show the robustness of our proposed protocols.

The remainder of this paper is organized as follows. Section 2 describes briefly the two studied protocols. We present in detail the Algorithms of these two protocols and their drawbacks in section 3. Our contributions are described in section 4. Section 5 concerns the simulations results. Section 6 concludes the paper.

2. State of Art

Real-time networks/protocols can be classified into three classes:

- Field buses [1, 3, 4, 5, 11, 12], token protocols [2, 7].
- Multiple access networks/protocols [8, 11, 12, 13].

We will focus here on the presentation in details of two real-time protocol belonging to the second class.

2.1. IEEE 802.5 Protocol

The IEEE 802.5 MAC protocol [7] defines over a ring structure, the arbitration of access based on a token

passing mechanism. Any station *x* that captures the token, transmits a data frame destined for a station *y* and *x* is the station that removes the data frame from the ring prior to regenerating the token.

2.2. The Timed Token Protocol

With the TTP [2], messages are distinguished into two types. Synchronous messages, such as voice and video transmission, are periodic messages which come to the system at regular intervals and have delivery time constraints. Asynchronous messages are non periodic messages which have no time constraints. In network initialisation time, all stations negotiate a common value for the Target Token Rotation Time (TTRT). Each station is assigned a fraction of the TTRT, known as its synchronous capacity, which is the maximum time for which the station is allowed to transmit its synchronous messages every time it receives the token. Whenever a station receives the token, it can transmit its synchronous messages, if any, for a time period which is not greater than its allocated synchronous capacity. The asynchronous messages can then be initiated, but only if the token has rotated sufficiently fast that it is "ahead of schedule" with respect to the TTRT. Such a protocol has been incorporated into many network standards, including the Fiber Distributed Data Interface (FDDI), the High Speed Data Bus and the High Speed Ring Bus (HSDB/HSRB), and the Survivable Adaptable Fiber optic Embedded NETwork (SAFENET).

3. The Studied Protocols

3.1. The Reservation Protocol (RP) of IEEE802.5

To be able to support different classes of network traffic, IEEE802.5 has the option of assigning priorities to messages. Each data frame and token contain three bits of priority, thus providing eight levels of priority. A node may only seize the token if it has data to transmit at a priority level, which is at least equal to the priority of the frame currently, circulating in the network. And all data transmitted during the capture of the token, may only be of equal or higher priority than the priority of the token. In addition, each data frame and token have a field indicating the highest priority of any node in the ring. These reservation bits are inspected each time the frame passes a node. If this node has data to transmit with a higher priority than the current reservation, the rerservation bits are updated to match the priority of the data the node wants to send. In this way the node requests the next token to be issued with a priority equal to the priority of the pending data at the head of its queue. When the token returns to the node having the highest priority, it may access the network medium and transmit its data. Formally the protocol uses the following variables:

- *P (Token)*: Stands for the current priority of the token.
- *P (Last_Token)*: Stands for the priority of the last token captured by the station.
- *P (reservation)*: Stands for the reservation priority of the token.
- *P (message)*: Stands for the highest priority among the priorities of messages enqueued by the station.
- *S (Token)*: Stands for the state of the token (free or busy).

Thus, the Algorithm is:

```
For each station, at the arrival of the token do:
  While there are messages in wait:
    If P(Token)≤P(message) and S(Token)=« Free » Then
     S(Token)=«Busy»;
     While P(message)≥P(reservation)Do
        P(Token)=P(message)
        Transmit message;
        Re-evaluate P(message);
     EndWhile
P(Token)=Max {P(Last_Token),   P(Reservation)};
S(Token)=« Free » ;
   Else
      If P(message)>P(Reservation) THEN
          P(Reservation)=P(message);
      EndIf
   EndIf
End
```

3.1.1. Critics

Lower priority packets may experience starvation, which is caused by large amounts of higher-priority packets waiting in the nodes queueus.

3.2. The Timed Token Protocol

This protocol already introduced in section 2 uses the following parameters:

- T_{TRT} (Target Token Rotation Time) defines the target rotation time of the token.
- $H_{k,\ k=0..m-1}$ (Synchronous capacity of node *k*), where *m* is the number of the stations in the ring. This parameter represents the maximum time for which a station is allowed to transmit synchronous messages every time it receives the token. Note that each station can be assigned a different H_i value. In this paper, we assume that $H_i=H_k,\ i\neq k\ \forall i,k \in \{0,..,m-1\}$.
- TRT_k (Token Rotation Time). It evaluates the cycle time this counter is initialized to the T_{TRT} value and re-initialized to this value either when the token arrives early to the station or when the TRT_k is expired.
- LC_k (Late Counter of node *k*). This counter is used to record the number of times that TRT_k has expired since the last token arrival at node *k*.
- THT_k (Token Holding Time), defines the time during which the station *k* may transmit non real-

time traffic.

Formally, the protocol is:

```
For each station k, (k=0, 1, 2,...m-1):
  THTk←0; LCk←0; TRTk←TTRT; /*initialization */
    Starting the countdown of TRTk
While the network is working:
    If TRTk=0 then
        TRTk←TTRT
        LCk←LCk+1
    EndIf
At the arrival of the token do:/* transmission */
        Case
            • LCk=0:  /* token early arrival case */
              THTk←TRTk
              TRTk←TTRT
              Starting the countdown of TRTk
              Transmission of real time messages during Hk
              Starting the countdown of THTk
              While THTk>0 and (∃ non real-time messages
                  in wait):
                  Transmission of non real-time messages
              Token passing to the station (k+1) (modulo m)
            • LCk=1: /* token late arrival case */
              LCk←0
              Transmission of real-time messages during Hk
              Token passing to the station (k+1) (modulo m)
            • LCk>1: /* error case */
                    « Error recovery» procedure
   EndCase
End
```

3.2.1. Critics

The "timed token" protocol presents a drawback towards asynchronous traffic. Indeed, if all the stations of the ring have permanently real-time (synchronous) and non real-time (asynchronous) messages, the first station transmits non real-time messages during T_{TRT}. Thereafter, only the real-time messages will be transmitted until at least one of the stations does not use all its synchronous capacity H_k to transmit the real-time traffic.

3.2.2. The Improved Timed Token Protocol

We improved in earlier work the TTP [2]. The main idea of our Improved Timed Token Protocol (ITTP) is to exploit the remaining time from H_k for the transmission of non real-time messages. The proposed protocol uses the same variables as those of the TTP with the introduction of a new variable HR_K that denotes the remaining time from H_k, after the transmission by the station k of all its real-time messages.

Principle*:*

- We assign to each station a time capacity H_k, that transmit the synchronous traffic.
- This protocol allows a station to transmit non real-time traffic whether:
  - It receives the token early.
  - It has an HR_k such that: $HR_k > 0$.

The second principle allows the transfer of asynchronous messages (if any) of the current station, instead of sharing its remaining time HR_k with the other stations, or waiting for the next token passing to send non real-time messages.

3.2.3. Critics

Let us consider the following critical situation. The first station has only non real time traffic and the other stations have real-time and non real time traffic. So, the first station transmits its traffic during (H_k+T_{TRT}). Greater is the number of the stations in the network, greater is the value of (H_k+T_{TRT}), and greater will be the number of the transmitted non real-time messages. On the other hand, an error situation (loss of the token) can arise if all stations have traffic to transmit during all their bandwidth H_k, as shown in Figure 1.

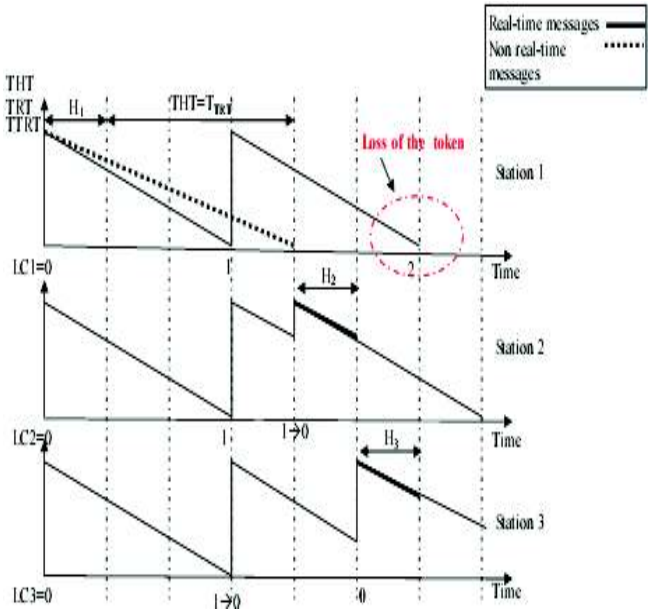


Figure 1. Illustration of a critical case running the improved timed token protocol [1].

According *t* Figure 1, at the first token arrival, the first station transmits all its non real-time traffic during H_1+T_{TRT} and then passes the token to the second station. The later will then transmits its reral-time traffic during H_2 and passes the token to the third station. At receipt of the token, the third station will transmit its real-time traffic during H_3 and passes the token to the first station. Unfortunately, in this situation, the token is lost and LC_1=2 (!).

4. Our Contributions

To overcome the drawbacks of the two studied protocols, we propose hereafter the following improved versions.

4.1. The Improved Reservation Protocol of IEEE802.5

To solve the problem of starvation of low priority messages, we propose the following changes:

- Our proposed protocol keeps the same principle and the same variables as those of the original algorithm, except the variable *P(last_token)*.
- We introduce a new variable *P(min)* which stands for the lowest priority among the priorities of messages waiting in the queue of a given station.

Thus, our Improved Reservation Protocol (IRP) of IEEE802.5 is the following:

For each station, at the arrival of the token do:
While there are messages in wait:
If P(Token)≤P(message) et S(Token)= «Free» Then
S(Token)= «Busy» ;
While P(message)≥P(Reservation)Do
P(Token)= P(message)
Transmit message;
Re-assess P(message);
EndWhile
P(Token)= P(Reservation) ;
S(Token)= « Free » ;
P(Reservation)= P(min) ;
Else
IF P(message)>P(Reservation) THEN
P(Reservation)= P(message) ;
EndIf
EndIf
End

To illustrate the robustness of our proposed protocol, let us take an example of a critical case using the resevation protocol of IEEE802.5. Two stations must transmit messages M_i=(M_1, M_2) having priorities P_i= (10, 20). Suppose that the first station sends a message of priority P_2 (lowest priority) and just after (before capturing the data frame of the first station) the second station will hold a message of higher priority P_1. Station 2 then assigns P_1 to *P(reservation)* and the station 1 will put *P(token)=P(reservation)=*P_1 before regenerating the token. The second station will assign the priority P_1 to *P(Last_token)*, and at the end of transmission, P_1 will be assigned to *P(token)*. Thus, *P(token)* will no longer have a value less than P_1. So, only messages of priority P_1 will be sent, and message of priority P_2 of station 1 will never be transmitted. The following table summarizes the progress of this algorithm (the characters in bold represent actions of station 2).

According to Table 1, the message of station 1 having the priority P_2 will never be transmitted. Let us now resolve this situation using our approach. The following Table 2 summarizes the progress of this Algorithm using our improved protocol (the characters in bold represent actions of station 2).

Table 1. Illustration of a critical case using the reservation protocol of IEEE802.5.

Steps	P (Token)	P (Reservation)	P (Last_ Token)	S (Token)	Station 1	Station 2
0	P2	*	P2	Busy	*	P1
1		P1				
2	P1					
3				Free		
4			P1			
5				Busy	P2 ?	*
6	P1					
7	P1					
8				Free		

Table 2. Resolving the previous critical case using the improved reservation protocol of IEEE802.5.

Steps	P(Token)	P(Reservation)	S(Token)	Station 1	Station 2
0	P2		Busy	*	P1
1		P1			
2	P1				
3			Free		
4		P3			
5			Busy	P2	
6	P1				
7		P2			
8	P2				
9			Free		
10		P3			

At the end of transmission of the higher priority message P_1 described before, the station 2 will set *P(reservation)=* P_3 (the lower priority). The second run of the protocol will give access to the message of the priority P_2 of station 1, since $P_2 \geq$ *P (reservation)=* P_3. So, the emission of a higher priority message doesn't prevent the low priority messages to be transmitted.

4.2. The Proposed Timed Token Protocol

The proposed protocol uses the same variables as those of the ITTP [2].

Principle:

- The protocol allows a station to transmit non real-time traffic whether it receives the token early or it doesn't use all of its allocated time capacity H_k. In addition, to overcome the weaknesses of the ITTP [2], we introduce these changes.

Case LC_k*=0:*

- Transmit real time traffic during H_k time units, and use the remaining time HR_k (if any) to transmit non real time traffic.
- Transmit non real time traffic during a time *THT*, but not exceeding H_k, this prevents the occurrence of the error: LC_k >1.

Case LC_k=1:

- Transmit real time traffic during H_k time units.
- If H_k is not expired, allocate the remaining time HR_k to transmit non real time messages.

4.3. Discussion

On one hand, with the RP of IEEE802.5 the low-priority messages may suffer from the starvation caused by large amounts of enqueued the high-priority messages. On the other hand, our proposed RP of the IEEE802.5, allows a fair transmission of both low and high priority messages. In addition to that, the number of higher priority messages is always greater that that of low-priority messages. On one hand, our improved timed token protocol [2] suffers from the high number of enqueued non real-time messages. This will lead for the transmission of more non real-time messages than real-time messages (!) On the other hand, our proposed timed token protocol allows the transmission of non real-time messages without waiting the next token arrival and with respect of the real-time constraint. The following simulations will confirm these assertions.

5. Simulations

We developed a tool in C language in order to evaluate the performances of our two improved protocols, on a Pentium 4 (2.88GHz), with 512Mb of memory.

5.1. The Improved Reservation Protocol of IEEE802.5

The aim of these first experiments is to compare both the reservation protocol of IEEE802.5 and our improved version. For this, we simulated a network where the number of stations varies from 5 to 100, during 1000 time units. Eight possible levels of priorities will be assigned randomly to the messages. If the priority of a message is greater than 5 then this message is of high priority, else it is of low priority. The stations queues can hold up to 50 messages. We consider here that there are always messages to transmit. The following graph illustrates the number of transmissions of high and low priority messages while varying the number of stations from 5 to 100 using the reservation protocol of IEEE802.5

According to the previous Figure 2, we note that the number of high priority messages is largely higher than the number of low priority messages. Indeed, according to this protocol, when a station sends a message of high priority, this priority will be assigned to the reservation priority of the token, which prevents the transmission of low priority messages. Let us now see the behavior of our approach facing the problem of transmission of low priority messages.

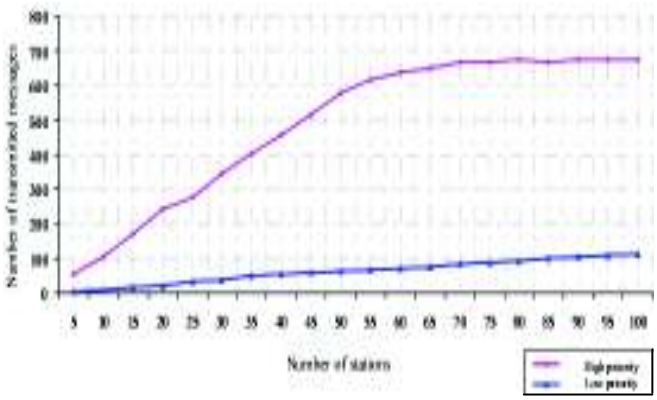


Figure 2. Number of transmitted messages using reservation protocol of IEEE802.5.

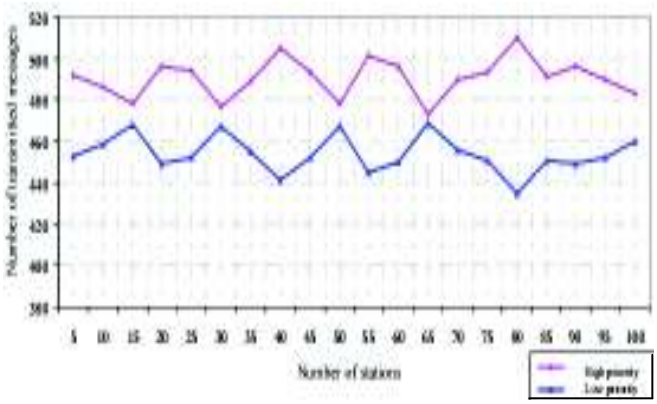


Figure 3. Number of transmitted messages using the improved reservation protocol of IEEE802.5.

Figure 3 shows a significant transmission of low priority messages explained by the fact that our protocol allows the transmission of these messages in case of the absence of higher priority messages. On the other hand, there are always more high priority messages transmitted than the low priority messages. So, the priority constraint of the messages is respected. According to Figures 2 and 3, even if the reservation protocol of IEEE802.5 allows the transmission of more high priority messages than our approach, nevertheless our protocol largely outperforms the former in term of the number of low priority messages transmitted. On the other hand, the total number of transmitted messages is higher using our protocol. In the next experiment, we will show the influence of the queue size of the stations on the number of transmitted messages (low and high priority messages). We consider here a network composed of 50 stations, and the queue size of each station varies from 50 to 5, and the priority of messages varies from 8 to 1.

Figure 4 shows that our protocol outperforms the reservation protocol of IEEE802.5 in term of the number of transmitted messages, especially when the queue size decreases from 25 to 5. Indeed, in case of the later protocol, when the queue size is small, it becomes quickly saturated by messages of low priority, since they can not be transmitted; this prevents also the transmission of high priority messages. On the other hand, our protocol allows the transmission of

practically the same number of messages (≅ 1000), because it allows a fair transmission of the two types of messages.

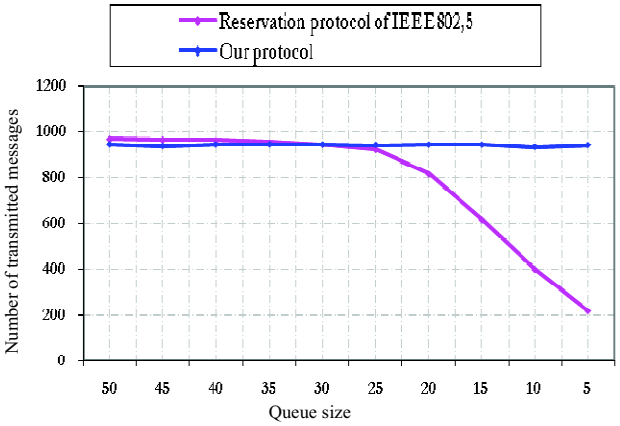
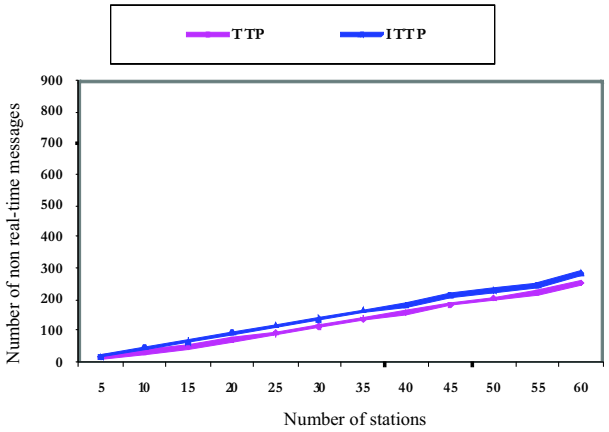


Figure 4. Number of total transmitted messages while varying the queue size.

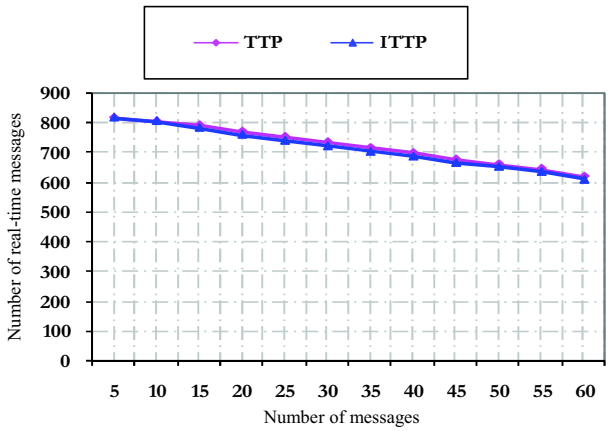
5.2. The Proposed Timed Token Protocol

First of all, we recall that we did not made any simulation in [2] to compare the TTP with our ITTP. This is the reason why we firstly present hereafter some experiments to compare these two protocols. After that, we will compare the ITTP to our new proposed timed token protocol. The aim of this first experiment is to compare the number of transmitted messages using the TTP and the ITTP while varying the number of stations from 5 to 60 during 1000 time units. The messages are randomly generated. We varied H_k from 5 to 3, and have considered the average results obtained from the three possible values of H_k.

On one hand, according to Figure 5-a, the ITTP lightly outperforms TTP in term of the number of non real-time messages transmitted while varying the number of stations from 5 to 60. On the other hand, Figure 5-b shows that both TTP and ITTP allows the transmission of practically the same number of real-time messages. In the next experiment we will look for the impact of the absence of real-time traffic on the transmission of non real-time traffic, using both TTP and ITTP.



a) Number of non real-time messages.



b) Number of real-time messages.

Figure 5. The number of (real-time/non real-time) messages transmitted by TTP and ITTP.

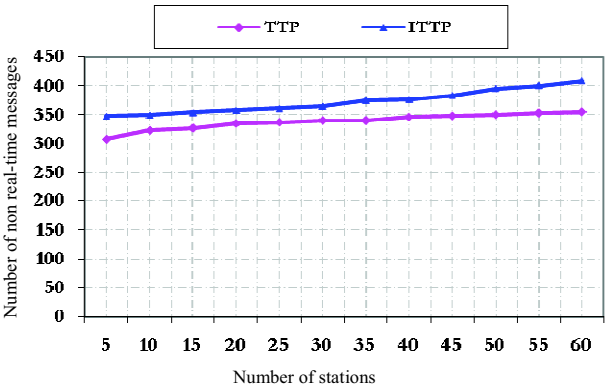


Figure 6. The number of non real-time messages transmitted in the absence of real-time traffic.

Figure 6 shows that ITTP allows the transmission of more non real-time messages than the TTP in the absence of real-time traffic. In the following we will show the impact of the TTRT parameter on the transmission of real-time messages using the ITTP. We simulate this case by varying the number of stations from 60 to 5, during 1000 time units, H_k here is such that H_k=5, and TTRT varies from 299 to 24.

Figure 7 shows that the number of real-time messages decreases when the number of stations allowed to transmit decreases. On the other hand, the number of non real-time messages increases. Indeed, each station k which does not have any message to transmit passes the token to the next allowing the token to arrive early to the latter. The latter station can then transmit real-time messages during H_k and non real-time messages during (HR+THT) which can reach its threshold value $TTRT$. So, the time allocated for the transmission of non real-time messages will be largely greater than the time allocated for the transmission of real-time messages. This explains the increase of the number of non real-time messages and the decrease of the number of real-time messages. Let us now see the impact of the TTRT parameter on the transmission of real-time messages using our proposed timed token protocol.

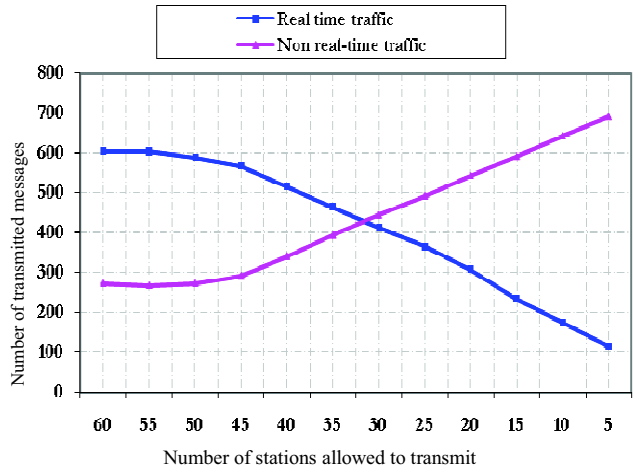


Figure 7. The influence of TTRT on the ITTP.

According to Figure 8, the number of real-time messages is largely greater than the number of non real-time messages. When the number of stations allowed to transmit decreases, the number of real-time messages decreases and the number of non real-time messages increases. This fact can be explained by the early arrivals of the token to the stations (as already seen with the ITTP), which allows the stations to transmit real-time messages during H_k and non real-time messages during at most H_k time units. In the worst case (the number of stations varies from 20 to 5), the number of the two types of messages is the same the two graphs coincide.

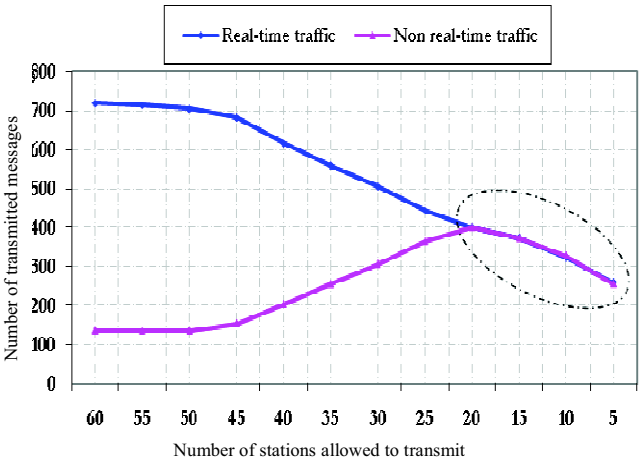


Figure 8. The influence of TTRT on the proposed timed token protocol.

According to Figures 7 and 8, we can conclude that our proposed protocol allows the transmission of more real-time messages than ITTP. It also allows the transmission of non real-time messages but without exceeding the number of real-time messages (with respect to real-time constraints).

6. Conclusions

Real-time LANs are gaining popularity in many embedded real-time systems because they can help to improve a system's scalability and reliability. We have identified three classes of theses LANs: fieldbuses, token protocols, and multiple access networks/protocols. We have highlighted drawbacks of two protocols of the second class: the reservation protocol of IEEE802.5, and the TTP. In case of the reservation protocol of IEEE802.5, the transmission of high priority messages may prevent the transmission of low priority messages. We resolved this problem by proposing an appropriate protocol allowing the transmission of higher and lower priority messages in adequate manner. Even if we improved in prior work the TTP our ITTP suffers also from a drawback. The new proposed TTP solves this drawback by allowing a station to transmit its non real-time traffic without waiting the next token arrival and during at most H_k time units. As future work, we plan to study another set of real-time LANs.

References

[1] Beghdad R., "Improving the Profibus Medium Access Control Protocol," *European Journal of Automated Systems*, vol. 40, no. 8, pp. 867-884, 2006.

[2] Bouzida Y., and Beghdad R., "Improving the Timed Token Protocol," *in Proceedings of the 1st IEEE International Conference of Networking*, UK, vol. 1, pp. 520-529, 2001.

[3] EN 50170, General Purpose Field Communication System, vol. 2/3, CENELEC, Brussels, 1996.

[4] EN 50170, General Purpose Field Communication System, vol. 1/3, CENELEC, Brussels, 1996.

[5] FIP Web Site, available at: http://www.worldfip.org/, last visited 2011.

[6] IEEE802.3, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD)," IEEE802.3 Standard, 1985.

[7] IEEE802.5, (ISO/IEC 8802-5:1998) IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 5: Token Ring Access Method and Physical Layer Specifications, 1998.

[8] Le-lann G. and Rivière N., "Real-Time Communications over Broadcast Networks: the CSMA-DCR and the DOD-CSMA-CD Protocols," *Technical Report*, Institut National de Recharche en Informatique et en Automatique INRIA, France, 1993.

[9] SAE J1583, *Controller Area Network CAN, an in-Vehicle Serial Communication Protocol*, SAE-Handbook, 1992.

[10] Vasques F., "Integration of Scheduling and Communication Mechanisms into the MAC Sub-

Layer of Real-Time Local Area Networks," *PhD Thesis*, Computer Science, LAAS, France, 1996.

[11] Venkatramani C. and Chiueh T., "Supporting Real-Time Traffic on Ethernet," *in Proceedings of 15th IEEE Real-Time Systems Symposium*, San Juan, pp. 282-286, 1994.

[12] Venkatramani C. and Chiueh T., "Design, Implementation and Evaluation of A Software Based Real-Time Ethernet Protocol," *in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, USA, pp. 27-37, 1995.

[13] Venkatramani C., "The Design, Implementation and Evaluation of RETHER: A Real-Time Ethernet Protocol," *PhD Thesis*, State University of New York at Stony Brook, 1997.

Sonia Mammeri obtained Master degree in computer sciences from the University of Bejaia in 2009. His research topic focuses on securing wireless networks.

Rachid Beghdad received his computer science engineering degree in 1991 from the ENITA School of Engineers, Algeria. He received his MSc computer science degree from Clermont-Ferrand University, in France 1994. He earned his PhD computer science degree from Toulouse University, in France 1997. He obtained his habilitation from the University of Constantine, 2010. He was the head of the computer science systems Laboratory in EMP school of Engineers. He was also the head of the scientific council of the department of computer science, of Bejaia University. He is currently, responsible of PhD graduate students in computer science at Bejaia University. He also leads and teaches modules at both BSc and MSc levels in computer science and computer networks. He is a reviewer for some journals, such as the Advances in engineering software journal, Elsevier, UK, the Computer Communications journal, Elsevier, UK, the WESEAS transactions on computer journal, Greece, and the IJCSSE journal, UK. He was also a reviewer for the CCCT'04, CCCT'05, CCCT'09, and CCCT'10 International Conferences, USA. His main current interest is in the area of computer communication system including intrusion detection methods, wireless sensor networks, unicast and multicast routing protocols, real-time protocols, and wireless LAN protocols.