Proc. of the 10th IEEE Int. Conference on Emerging Technologies and Factory Automation (ETFA 2005), Catania, Italy, Sept. 2005.

A Distributed Architecture for Mobile Robots Coordination [*]

Gianluca Franchino, Giorgio Buttazzo, Tullio Facchinetti
University of Pavia, Italy
Email: {gianluca.franchino, giorgio.buttazzo, tullio.facchinetti}@unipv.it

Abstract

In many application fields, such as surveillance, monitoring, exploration, and rescuing, the use of multiple coordinated robot units seems to be the most convenient solution, in terms of costs, performance, and efficiency. In this paper we present a distributed real-time architecture for coordinating a set of mobile robot that have to operate according to a common objective. The interaction with the environment causes each unit to operate under real-time constraints, which are enforced by the operating system to achieve a desired level of performance. The design of the architecture has been focused on three main aspects: real-time communication, modularity, and flexibility. The last two aspects are essential in a distributed environment to simplify maintenance and system upgrades. The proposed approach has been implemented on a team of six robot vehicles, equipped with proximity and light sensors, that can operate according to predefined formations.

1 Introduction

Coordinating a team of mobile robots requires several difficult problems to be solved, including robot localization, real-time communication, schedulability analysis of real-time tasks and messages, and efficient resource management (e.g., for optimizing energy consumption).

All of these issues have been addressed in the real-time literature, but a complete and robust solution to them is not yet available. Robot localization is one of the hardest problems, for which different approaches have been proposed, including the use of navigation maps through which a robot can get its position. In unknown environments, however, active exploration is required to build the map incrementally. The disadvantage of this method is that the time needed to build the map can be too long.

When the robots operate in close vicinity, self-localization can be achieved by determining the relative positions with respect to the other robots. Additional information can be taken from the environment through local reference points, like known objects or beacons, or absolute reference points, like the sun, the stars and magnetic north. The use of multiple reference points allows a robot to determine its position by means of triangulation. Silva, Santos and Almeida [17] proposed a self-localization method that combines absolute azimuthal information from an analogue compass with relative odometric information, obtained from an optical mouse.

[*]This work has been partially supported by the Italian Ministry of University Research under contract 2003094275 (COFIN03) and contract 2004095094 (COFIN04).

The interactions with the environment and with the other robots impose stringent timing constraints on the robot control activities that need to be enforced by adopting suitable kernel mechanisms. In a team of cooperating robots, the scheduling problem becomes even more complex due to the communication messages exchanged in the team. As a consequence, a precise estimation of the end-to-end communication delays is necessary to provide any guarantee on the robot performance.

Li et al. [11] proposed two heuristic algorithms to allocate real-time tasks to the various execution units in the team. Both methods consider the communication cost and the utilization bound to allot the tasks at the processors and to guarantee their schedule. Shi et al. [16] described a robotic system for the automatic deployment of highway safety markers. The architecture consists of one lead robot (the foreman) and a group of guided robotic safety markers (the robotic barrels), which are guided by the foreman. The system consists of two classes of real-time activities, modelled as periodic tasks and variable rate execution tasks [9], and schedulability analysis is performed under the Earliest Deadline First (EDF) and the Rate Monotonic (RM) algorithms [12]. Whereas in [16] the authors only considered the problem of detecting the robotic barrels and controlling them by sending appropriate set points, in a successive work Qadi et al. [14] analyzed the foreman navigation, the accuracy placement of the barrels, and the relations between the periods of these tasks and the robot performance, i.e., the navigation velocity.

In a distributed system for real-time applications, the key of success is the timely execution of the processes running in the system's nodes, which communicate to achieve a common goal. This cannot be achieved without a proper support of a network layer that provides a timely message transmission. The communication among mobile units requires a careful management of the transmission medium (typically a radio channel) for achieving a sufficient level of predictability on messages exchange. Full autonomy of the robotic team can only be obtained through a wireless ad-hoc network [18]. It is worth noticing that when robots have weak hardware resources, (e.g., slow processors and low power radio transceivers), such physical constraints need to be taken into account in the network design.

Facchinetti et al. [7] proposed a MAC level protocol, based on implicit EDF [4], that allows nodes to enter and leave the communication area, avoiding collisions due to simultaneous transmissions. This protocol was then used to coordinate a small team of robots cooperating for a common goal [6].

In this paper we report our experience and solutions to the problems we encountered in the development of a

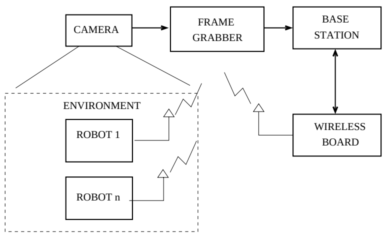

Figure 1. Block diagram of the system.

team of autonomous vehicles. In particular, we focus on the problem of scheduling a set of tasks and messages with stringent time constraints, and we show how the use of a real-time kernel is essential for achieving high predictability and simplifying the performance analysis of the system. Moreover, we describe the communication system that supports the real-time messages exchange among the tasks running on the various computational nodes.

The rest of this paper is organized as follows. Section 2 describes the system architecture, presenting the robot vehicles, the localization method, and the control scheme. Section 3 illustrates the communication system. Section 4 presents the real-time issues of the system, including task models and feasibility analysis for tasks and messages. Section 5 reports some experimental results, and Section 6 states our conclusions and future work.

2 System architecture

The system consists of a set of robotic vehicles actuated by two independent wheels and equipped with proximity and light sensors. Robot localization is achieved by simulating a global positioning system (GPS) through a fixed camera located above the environment. A personal computer, acting as a base station, processes the images coming from the camera to identify the position and the orientation of each robot. The role of the base station is to interact with a human operator and execute a given control strategy by sending proper commands to the robots.

The robots and the base station communicate through a wireless board, which allows sending messages at a speed of 38400 bits per second. A block diagram of the system architecture is illustrated in Figure 1. The communication software is structured in two hierarchical levels. The low-level software runs on the wireless board on a dedicated programmable interrupt controller (PIC); it manages the radio channel and sends the messages coming from the higher level. The high-level software runs on the robot micro-controller and on the base station. The communication system will be described in detail in Section 3.

2.1 Localization

In order to concentrate on real-time and communication issues and perform some real experiments on the system, we decided to simplify the localization problem by simulating a global positioning system (GPS) through a videocamera mounted above the robot workspace, acquiring images from the environment at a frequency of 25 frames per second. Images are sent to the base station, which processes them to identify the robots and compute their positions and orientations. To simplify the recognition process, each robot is uniquely identified by means of proper markers, each realized by two colored disks.

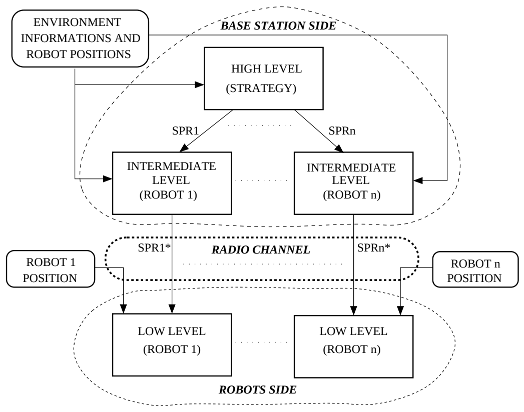

Figure 2. Hierarchical control scheme.

2.2 Robot control

The control architecture of the system is organized in three hierarchical levels, as illustrated in Figure 2. The higher level is responsible for the coordination of the team and runs on the base station. It executes the strategy to plan the actions that each robot has to carry out in order to accomplish the common mission. It receives as inputs the robot positions computed from the camera and the sensory data coming from the robot vehicles, which directly detect the presence of obstacles along the path. It provides as outputs the set points $(SPR_1, \ldots, SPR_n)$ that each vehicle must reach.

The middle control level (also running on the base station) receives as inputs the set points computed by the upper level and the sensory data from the robots. The purpose of the middle level controller is to decompose the set points provided by the upper level into a sequence of intermediate set points, which can be used, if necessary, to plan an alternative path to reach the location provided by the higher control level. In other words, the goal of the high-level controller is to plan the final location of each robot, whereas the middle-level controller decides the specific path to be used to reach that location. The goal of the lower control level (running on each robot) is to properly drive the robot wheels to reach the set points provided by the middle control level. A classical PI (Proportional and Integral) controller is used at the lowest level to reduce the position error of the vehicle.

3 The communication system

A key issue for achieving a good performance in a team of mobile robots is an efficient and predictable communication infrastructure that allows robots to exchange real-time messages with bounded delays. Our communication system consists of n computational nodes ($n-1$ robots and the base station), each including two devices, a DTE (Data Terminal Equipment), which can be either the robot's CPU or the personal computer (for the base station), and a DCE (Data Communication Equipment), which is a wireless board.

These two devices communicate through an RS232 serial line. The communication software is organized into three layers. The application level runs in the DTE and provides a communication service that follows a producer-consumer model, according to which the node that generates information (producer) starts the transaction, and the node that needs information (consumer) retrieves it from the network. The model is based on broadcast transmission (each message is received by all nodes). The other two software levels, data link and physical, run into the DCE. The data link layer is responsible for the addressing mode, the radio channel management, and the message format. A direct addressing is used, according to which each node has a unique address that identifies it in the network. There is also an address that identifies broadcast messages destined at every node. In order to manage the radio channel, we have developed a MAC protocol that guarantees the transmission of hard real-time messages with bounded delay. All the nodes are fully connected, therefore the network topology is a mesh.

3.1 The MAC protocol

The MAC protocol used in this system manages the access to the radio channel, guaranteeing the timing constraints of the messages through a timed token-based mechanism, first introduced by Grow in 1982 [10]. Since then, extensive research has been done on the timed token protocol, with particular emphasis on real-time issues [13, 19, 1]. In the remainder of this section, we present the protocol and describe our solutions to the problems we have encountered.

The timed token [13] is a token-passing protocol in which each node receives a guaranteed share of the network bandwidth. There is a token that travels between nodes in a circular fashion and each node can transmit only when it possesses the token; this guarantees a collision-free medium access. The key idea is to assign each node a time budget, which is the maximum time the node is permitted to transmit messages every time it receives the token. The system starts by assigning the token to node 1, which can transmits its messages, if any, for a period no longer than its allocated time budget. Then, node 1 sends the token to node 2, which can transmit its messages, if any, and so forth until the token is passed to node n. When node n finishes its transmission, the token is sent back to node 1.

There are several budget allocation schemes, some of them have been analyzed in [1, 19]. For each scheme, the worst-case achievable utilization U^* (defined later in this section) has been derived so that, if the utilization of the periodic message set is less than or equal to U^*, the scheme guarantees that message deadlines are met. We adopt an allocation scheme with $U^* = 0$, hence it is not possible to use the worst-case achievable utilization to guarantee messages deadlines. The method for guaranteeing messages constraints is explained in the next section.

Notice that our network is fully connected as far as message delivery is concerned, but circular from the token circulation point of view. There is an important protocol parameter, called the TTRT (Target Token Rotation Time), which gives the expected token rotation period. Let σ be the time needed for transmitting the token between two nodes, including the overhead introduced by protocol. If H_i is the maximum transmission time assigned to node i, then TTRT can be computed as

$$TTRT = \sum_{i=1}^{n} H_i + n\sigma.$$

In terms of bandwidth, the token transmission overhead can be considered as a transmission of a message of length $n\sigma$ with a period equal to TTRT. Therefore, the total net bandwidth available for transmission is $1 - n\sigma$ / TTRT. The message stream S_i generated by each node i can be described by the tuple (C_i, T_i, D_i) where:

- C_i is the maximum amount of time required to transmit a message in the stream. This includes the time required to transmit both the payload data and the message headers.
- T_i is the interarrival period between messages in the stream. If the first message in stream S_i of node i is put in the transmission queue at time $t_{i,1}$, then the j-th message in stream S_i will arrive at node i at time $t_{i,j} = t_{i,1} + (j-1)T_i$, where $j \geq 1$.
- D_i is the relative deadline associated with messages in stream S_i, that is, the maximum amount of time that can elapse between a message arrival and completion of its transmission. Thus, the transmission of the j-th message in stream S_i that arrives at $t_{i,j}$ must be completed not later than $d_i = t_{i,j} + D_i$, which is the message's absolute deadline.

Messages of each stream can be sent either to a precise receiver or to all by means of broadcasting. The channel utilization of each message in the stream S_i is

$$U_i = \frac{C_i}{min(T_i, D_i)}.$$

The total effective channel utilization, U, of a periodic message set is then $U = \sum_{i=1}^{n} U_i$, which measures the total demand placed on the system by the entire periodic message set. The parameters described above are crucial for guaranteeing periodic messages deadlines. Any choice of these parameters must satisfy the following constraints:

1. **Protocol Constraint**: The total bandwidth allocated to the nodes must be less than the available network bandwidth, that is,

   $$\frac{\sum_{i=1}^{n} H_i}{TTRT} \leq 1 - \frac{n\sigma}{TTRT}.$$

   This constraint is necessary to ensure a stable operation of the timed-token protocol.
2. **The deadline constraint**: This constraint states that every periodic message must be transmitted before its absolute deadline. Formally, let $s_{i,j}$ be the time at which the transmission of the j-th message in stream S_i is completed. The deadline constraint requires that for $i = 1, \ldots, n$ and $j = 1, 2, \ldots$, $s_{i,j} \leq t_{i,j} + D_i$, where $t_{i,j}$ is the message arrival time and D_i is its relative deadline. Note that in the above inequality, $t_{i,j}$ and D_i are given by the application, but $s_{i,j}$ depends on the synchronous bandwidth allocation and on the TTRT value.

3.2 Message constraints guarantee

Our protocol scheme assigns each node a maximum time budget H_i sufficient to transmit an entire message, that is:

$$\forall i = 1, \ldots, n \quad H_i = C_i.$$

For this scheme it has been shown [1] that $U^* = 0$, meaning that message timing constraints cannot be guaranteed using this method. To guarantee real-time message delivery with this scheme, we provide a test on the messages relative deadlines D_i that, if satisfied, guarantees to meet all absolute deadlines of the periodic messages.

If the token arrives to each node with no delay, the maximum amount of time occurring between two consecutive token arrivals at a node is TTRT time units. Otherwise, Johnson and Sevciks [15] have shown that, if the token is late, the maximum amount of time that may pass between two consecutive token arrivals at a node is $2TTRT$. Furthermore, they have shown that the average rotation time is not greater than TTRT.

If a message comes in the i-th transmission queue at time k, it will be sent after $Tq_i(k)$ time units, where $Tq_i(k)$ denotes the maximum time that a message, arrived at time k on node i, can stay in the transmission queue. Hence, if for each i and for each time instant k $D_i \geq Tq_i(k)$, all message deadlines will be met.

Lemma 1 *For $i = 1, \ldots, n$, if $H_i = C_i$ and the token is not late, then*

$$Tq_i(k) = q_i(k)TTRT;$$

if the token is late, then

$$Tq_i(k) \leq q_i(k)(2TTRT + H_i)$$

where $q_i(k)$ is the number of messages in the transmission queue of node i at time k.

Proof. First we consider the case in which the token is not late. Let us suppose at time $(k-1)$ the queue is empty, and at time k a message comes. Also, let us suppose the token left node i at time $(k-\epsilon)$ with ϵ infinitesimal (i.e., the token left the node a moment before the message arrived) and let us suppose each other node has one or more messages in the transmission queue. Under these hypotheses the message in the queue of node i must wait the maximum time before its delivery. The next arrival time of the token to node i is $k + T_i^{delay}$, where $T_i^{delay} = \sigma + \sum_{j=1, j\neq i}^{n}(H_j + \sigma)$. So, after T_i^{delay} time units the token comes back to node i and the entire message can be sent because $H_i = C_i$. After $T_i^{delay} + H_i$ time units the message has been delivered. It is trivial to verify that, if the token is not late,

$$T_i^{delay} + H_i = \sigma + \sum_{j=1,j\neq i}^{n}(H_j + \sigma) + H_i = H_i + \sigma + \sum_{j=1,j\neq i}^{n}(H_j + \sigma) = \sum_{j=1}^{n}(H_j + \sigma) = TTRT.$$

If at time $(k-1)$ there are p messages in the queue of node i and at time k a new message arrives, then in the worst case $q_i(k) = p+1$ and the token left the node at $(k-\epsilon)$. Therefore, after T_i^{delay} time units, the token comes back to the node and an entire message is delivered. Then, at time $t = k + T_i^{delay} + H_i = k + TTRT$, $q_i(t) = p$ and the token leaves node i. Again, after T_i^{delay} time units the token comes back and an other message can be sent. Then, at time $t = k + T_i^{delay} + H_i + T_i^{delay} + H_i = k + 2\cdot(T_i^{delay} + H_i) = k + 2\cdot TTRT$, $q_i(t) = p-1$. It follows that, at time $k + sTTRT$, s messages have been sent. For $q_i(k) = p+1$, the $(p+1)$-th message comes into the queue at time k and leaves the queue after $Tq_i(k) = (p+1)TTRT = q_i(k)TTRT$.

If the token is late, then $max(T_i^{delay}) = 2TTRT$, therefore, if $q_i(k) = 1$, at time $k + 2TTRT$ the token comes back to the node and the message can be transmitted, then $Tq_i(k) = 2TTRT + H_i$. Following the same reasoning in the first case (token not late), if $q_i(k) = p+1$, the message comes into the queue at time k and, in the worst case, leaves the queue after $Tq_i(k) = q_i(k)(2TTRT + H_i)$ time units. $\Box$

The following lemmas provide an upper bound to $q_i(k)$ and $Tq_i(k)$.

Lemma 2 *For any i $(i = 1, \ldots, n)$, for any $k \geq 0$, if $H_i = C_i$, $T_i \geq TTRT$, and the token is not late, then $max(q_i(k)) = 1$.*

Proof. The arrival period of the transmission queue i is equal to T_i. As shown in Lemma 1, if the token is not late, the emptying period of the transmission queue i is equal to TTRT. For $k = 0$, $q_i(k) = 1$, for $k > 0$, $q_i(k) = 1 + \lfloor \frac{k}{T_i} \rfloor - \lfloor \frac{k}{TTRT} \rfloor$. And since $T_i \geq TTRT$, $\lfloor \frac{k}{T_i} \rfloor - \lfloor \frac{k}{TTRT} \rfloor \leq 0$. Therefore $q_i(k) \leq 1$. $\Box$

Lemma 3 *For any i $(i = 1, \ldots, n)$, for any $k \geq 0$, if $H_i = C_i$ and $T_i \geq 2TTRT + H_i$, then $max(q_i(k)) = 1$.*

Proof. The arrival period of the transmission queue i is equal to T_i. As shown in Lemma 1, the emptying period of the transmission queue i is equal to $(2TTRT + H_i)$. For $k = 0$, $q_i(k) = 1$, for $k > 0$, $q_i(k) = 1 + \lfloor \frac{k}{T_i} \rfloor - \lfloor \frac{k}{2TTRT + H_i} \rfloor$. And since $T_i \geq 2TTRT + H_i$, $\lfloor \frac{k}{T_i} \rfloor - \lfloor \frac{k}{2TTRT + H_i} \rfloor \leq 0$. Therefore, $q_i(k) \leq 1$. $\Box$

Theorem 1 *For every i $(i = 1, \ldots, n)$, if $H_i = C_i$, $T_i \geq TTRT$, and the token is not late, then the stream set $\mathcal{M} = \{S_1, S_2, \ldots, S_n\}$ is schedulable if and only if $D_i \geq TTRT$.*

Proof. $\mathcal{M}$ schedulable means that both of protocol constraints are satisfied. Since $H_i = C_i$, then $TTRT = \sum_{i=1}^{n}(H_i + \sigma) = \sum_{i=1}^{n} H_i + n\sigma$. Hence, $\sum_{i=1}^{n} H_i = TTRT - n\sigma$, meaning that the **Protocol Constraint** is satisfied.

(Only if). To satisfy the **deadline constraint** a node must have at least a possibility to send each message before the message deadline expires, therefore $\forall i$ $D_i \geq TTRT$.

(If). From Lemma 2, when a message is queued at time k, $q_i(k) = \max(q_i(k)) = 1$, that is, it is the only message in the queue. Then, from Lemma 1, $Tq_i(k) = q_i(k)TTRT$, hence $Tq_i(k) = TTRT$. We remember that $Tq_i(k)$ is the time elapsed from the time the message arrived in the queue and the time it left. Therefore, if $D_i \geq TTRT$, $D_i \geq Tq_i(k)$, meaning that the **deadline constraint** is satisfied. $\Box$

Theorem 2 *For any i $(i = 1, \ldots, n)$, if $H_i = C_i$, $T_i \geq 2TTRT + H_i$, and the set $\mathcal{M} = \{S_1, S_2, \ldots, S_n\}$ is schedulable if and only if $D_i \geq 2TTRT + H_i$.*

Proof. The proof is similar to that of Theorem 1, considering that the minimum dimension of the transmission queue is given by Lemma 3 and, if the token arrives with the maximum delay, then $Tq_i(k) = 2TTRT + H_i$. $\Box$

In our system, the choice of a time-token protocol was motivated by its implementation simplicity and its low overhead. The time budget H_i allocated to each stream is $H_i = C_i$, which gives the possibility to each node to transmit an entire message every time it receives the token. Moreover, this scheme simplifies the message management, because the system must not manage different pieces of messages in order to rebuild the messages delivered. As demonstrated in [1], it is impossible to provide a guarantee on real-time messages based on U^*, because for this scheme $U^* = 0$. Furthermore, to the best of our knowledge, all allocation schema proposed for the timed-token protocol have $max(U^*) = \frac{2\cdot(1-\alpha)}{5+\alpha}$ [19], where $\alpha = \frac{n\sigma}{TTRT}$. As shown in Section 5, in our application the utilization of the message set is greater than $max(U^*)$. In order to guarantee that the deadline constraints are satisfied, we demonstrated that if the condition $D_i \geq TTRT$ is met, the soft real-time delivery of the messages is guaranteed, if the token is not late. Finally, we showed that, since the token delay is bounded [15], if $D_i \geq 2 \cdot TTRT + H_i$, the hard real-time delivery of the messages is guaranteed, even if the token is always late.

3.3 Robustness

When robots move, it is possible that one or more members of the team fail or leave the communication zone. In this case, the token would be lost. In order to guarantee a certain degree of robustness, when node j sends the token packet to node $j + 1$, it waits for an acknowledgement. Notice that during the waiting time the node is not blocked. If for k times the acknowledge does not come and the ring recovery process starts [5], node $j + 1$ is considered not connected to the network and the token from node j is sent to node $j + 2$. The parameter k is set at the system initialization. In this paper, the case in which a robot may rejoin the team is not considered and will be addressed in a future work.

4 Real-Time issues

Robot team performance, such as the robot continuous motion, robot coordination and obstacle avoidance, depends on the timely execution of the tasks running in every processing element of the architecture. In this section we will analyze the relations between tasks parameters (i.e., periods and deadlines) and performance parameters (i.e., robot speed). Moreover, we will describe our solution to the scheduling problems induced by this robotic application.

All tasks in the application are modeled as periodic or aperiodic. Each periodic task τ_i is identified by four parameters, (C_i, T_i, D_i, ϕ_i), where C_i is its worst-case computation time, T_i its period, D_i its relative deadline, and ϕ_i its offset, i.e. the arrival time of the first job. Aperiodic tasks are described by the same parameters, with the difference that T_i indicates the minimum interarrival time between two consecutive instances of the task. For aperiodic tasks, T_i is used to guarantee feasibility in the worst case activation scenario. Tasks can communicate both through critical sections accessed by the Stack Resource Policy [2], or by means of non blocking asynchronous buffers (CABs) [3]

To enforce timing constraints on the real-time activities running in the base station, tasks are executed by the Shark kernel [8], which is a configurable real-time kernel able to handle hard and soft real-time tasks under different scheduling algorithms, selectable by the user at system's initialization.

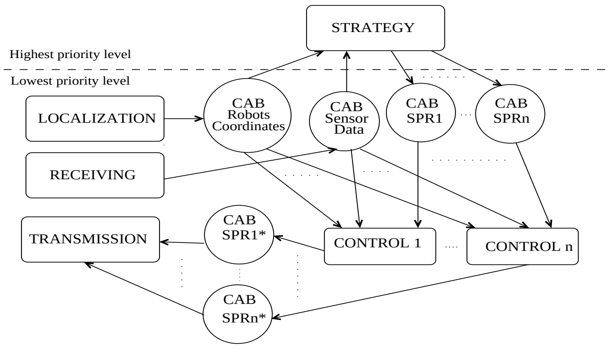

Figure 3. Tasks and resources interaction on the base station.

4.1 Software

As already shown in Section 2, part of the software runs in the base station and part in the robot units. More precisely, on the base station there is a periodic task which performs the localization algorithm (see Section 2.1), another that performs the coordination strategy phase, and $n - 1$ tasks (one for each robot) that execute the middle level control (see Section 2.2). There are also two periodic communication tasks responsible for exchanging messages between the DTE and the DCE in the base station (see Section 3).

In each robotic vehicle there are four tasks, one (τ_{lc}) for the low-level control (Section 2.2), one that collects data from the sensors (τ_{rs}), and two for message exchange (τ_{rr}, τ_{rt}), like the ones running in the base station.

4.2 Task parameters

The values assigned to the task parameters depend on the performance requirements of the system or are imposed by some hardware device. For example, the camera placed above the workspace (see Section 2) provides a frame every $T_l = 40$ msec; this means that the $Localization$ task has to execute with a period equal to T_l in order to update robot positions at video rate.

The period of the $Strategy$ task closely depends on the application. For example, if the strategy process has only to decide a few set points, the task can be executed with a relatively long period (e.g., one second). Viceversa, if the robotic team must be coordinated to catch a fast object, e.g., an enemy robot, the strategy task must run with a much shorter period (e.g., 40 msec).

The period of each task at the middle control level influences the continuous motion of the associated robot. Each vehicle used in our application has a pair of dc servomotors for driving the robot wheels. In order to achieve as smooth motion, each servo must receive a command every 40 msec. This affects the period T_c of the control task, at the middle level of each robot. Moreover, T_c also depends on the actual velocity of the associated vehicle; in fact, to avoid an obstacle, the control process has to decide the appropriate action enough in advance. The application on the base station is structured as illustrated in Figure 3.

4.3 Communication constraints

We remind that our system consists of $n = 7$ computational nodes, including one base station and six robot vehicles. Each middle level control task sends only relative positions to the related robot (i.e., the difference

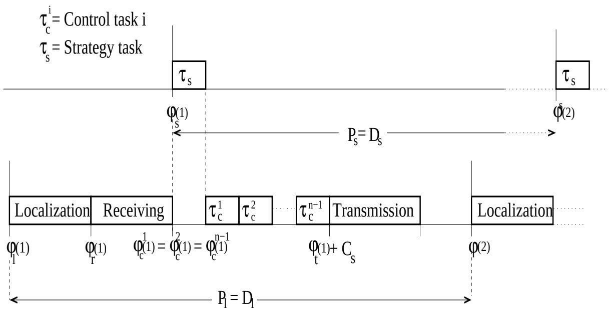

Figure 4. Schedule on the base station.

between the set point and the actual position). To send this information, the base station should deliver 6 messages, one for each robot. To decrease the number of messages exchanged, relative positions and orientations of every robot are inserted in a single broadcast message that is forwarded to all vehicles. The message length is equal to 41 bytes (36 for data payload), meaning that for the message stream S_b of the base station the message transmission time is

$$C_b = \frac{(41 * 10)bit}{38400\ bit/sec} = 10.67\ msec.$$

As explained in Section 3.1, the MAC protocol assigns this stream a time budget $H_b = C_b$.

The other nodes, the vehicles, have to send a message with the data coming from their onboard sensors. To do this, each robot has a message stream S_r associated to it. The length of this message is equal to 9 bytes (4 for data payload), meaning that the message transmission time is

$$C_r = \frac{(9 * 10)bit}{38400\ bit/sec} = 2.34\ msec.$$

As before, the assigned time budget is $H_r = C_r$.

The token rotation time (TTRT), in milliseconds, results to be

$$TTRT = H_b + (n-1) \cdot H_r + n \cdot \sigma = 10.67 + (n-1) \cdot 2.34 + n \cdot 1.04 = 8.33 + n \cdot 2.86.$$

Therefore, for $n = 7$, $TTRT = 31.99$ msec. From TTRT we can obtain the minimum period (T_i) and the minimum relative deadline (D_i) for each message stream S_i that the protocol can guarantee. Every message in each stream becomes obsolete every time (T_i) a new instance of the message comes; so we put $D_i = T_i$.

In order to guarantee the schedulability of messages delivery, the relative deadline of a stream S_i cannot be smaller than $2TTRT + H_i$, which is the maximum message transmission delay, as stated in Theorem 2. If a soft real-time guarantee is sufficient, then the relative deadlines cannot be smaller than $TTRT$, as stated in Theorem 1.

4.4 The scheduling solution

The tasks on the base station have both temporal and precedence constraints. The correct temporal sequence of the task schedule is shown in Figure 4.

The $Localization$ task computes the robot positions and orientations. Then, the $Receiveing$ task retrieves the messages the team vehicles have sent to the base station. These messages contain the robots sensory data. When the $Receiveing$ task finishes, the $n-1$ $Control$ tasks start in sequence; when the last $Control$ task (the n-th) finishes, the transmission buffer of the base station contains the message with the relative positions of the robots. Then, the $Transmission$ task transfers this message to the wireless board. The constraint of the $Strategy$ task is that $\phi_s(i) = C_l + C_r + (i-1) \cdot T_s \quad (i > 1)$ and its first instance must start before the first instance of the control tasks. To satisfy such a constraint, its priority level is set greater than the others and its offset is set to $\phi_s(1) = \phi_c^1(1)$, where $\phi_c^k(j)$ denotes the offset of the j-th job of the k-th control task. From the considerations presented above, the following task parameters can be derived:

1. $Localization$ task τ_l, $\forall i \geq 1$:

$$\begin{cases} \phi_l(i) = (i-1) \cdot T_l \\ D_l = T_l. \end{cases}$$

2. $Receiver$ task τ_r, $\forall i \geq 1$:

$$\begin{cases} \phi_r(i) = \phi_l(i) + C_l \\ D_r = T_r = T_l. \end{cases}$$

3. $Control$ task τ_c^k, $\forall i \geq 1$, for $k = (1, \ldots, n-1)$:

$$\begin{cases} \phi_c^1(i) = \phi_r(i) + C_r \\ \phi_c^k(i) = \phi_c^{k-1}(i) + C_c \quad (k > 1) \\ D_c^k = T_c^k = T_l. \end{cases}$$

4. $Transmission$ task τ_t, $\forall i \geq 1$:

$$\begin{cases} \phi_t(i) = \phi_c^{n-1}(i) + C_c \\ D_t = T_t = T_l. \end{cases}$$

5. $Strategy$ task τ_s, $\forall i \geq 1$:

$$\begin{cases} \phi_s(1) = \phi_c^1(1) \\ \phi_s(i) = C_l + C_r + (i-1) \cdot T_s \quad (i > 1). \end{cases}$$

In order to enforce the precedence constraints we used two hierarchical priority schedulers in the Shark kernel. The $Strategy$ task is handled by a RM scheduling module defined at the highest priority level, whereas all periodic tasks are scheduled by another RM module defined at a lower priority level. In this way, since all tasks at the lower level have the same period, they are scheduled in sequence, as shown in Figure 4.

The tasks running on the robot CPUs (τ_{lc}, etc.) are executed non preemptively. The ensemble of these tasks can be viewed as a single task τ_v, having $C_v = C_{rr} + C_{rs} + C_{lc} + C_{rt}$, and period $T_v \leq 40$ msec to satisfy the constraint imposed by the servomotors control.

The $Strategy$ task cannot be interrupted during its execution, because it executes at the highest priority level; then, if $C_s \leq D_s$, it always finishes before its deadline. The rest of tasks, those at the lowest level, execute in sequence and can be preempted only by the $Strategy$ task. Hence, the maximum delay they can suffer is equal to $C_s \cdot \lceil \frac{T_l}{T_s} \rceil$. Therefore, to guarantee a feasible schedule of the task set is sufficient to verify that

$$C_l + C_t + C_r + (n-1) \cdot C_c + C_s \cdot \lceil \frac{T_l}{T_s} \rceil \leq T_l. \quad (1)$$

Notice that, if $C_l + C_t + C_r + (n-1) \cdot C_c + C_s \cdot \lceil \frac{T_l}{T_s} \rceil \geq T_l$, being $D_t = D_l = T_l$, at least the execution of τ_t breaks its deadline. It follows that the condition is also necessary.

In order to maintain, for each vehicle, a continuous motion without crashing, the velocity of the vehicles and the period of sensor sampling and control tasks must be carefully chosen.

In the remainder of this section, we show the relation between the robot velocity v_r and the periods $T_l = T_c = T_r = T_t$ of the robot tasks in the base station. Let us make the following definitions:

- D_o be the distance of an obstacle from the robot;
- D_{safe} be the minimum distance from an obstacle that a robot has to maintain to ensure a safe brake;
- T_{samp} be the maximum sampling period of the robot sensors by the base station, given by $T_{samp} = T_v + C_{lc} + C_{rt} + 2\text{TTRT} + H_r + T_l$;
- T_{react} be the maximum interval from the time the system gets the information from the sensors and the time the robot carries out the next command. It is given by $T_{react} = C_s + (n-1) \cdot C_c + C^b_{tx} + 2\text{TTRT} + H_b + T_v + C_{rs} + C_{lc}$;
- $Q(v_r)$ be the braking space needed for a complete stop of the vehicle. As an example, assuming a brake entirely due to friction, $Q(V_r)$ is given by $Q(V_r) = \frac{v_r^2}{2\mu g}$, where g is the gravitational acceleration and μ is the friction coefficient.

Hence, in order to avoid a possible crash, the following condition has to be satisfied:

$$D_o \geq D_{safe} + v_r(T_{samp} + T_{react}) + Q(v_r).$$

To derive the relation between T_l, D_o and v_r, the interval $T_{react} + T_{sample}$ can be considered as the sum of T_l plus a constant term $k_l = T_{react} + T_v + C_{lc} + C_{rt} + 2\text{TTRT} + H_r$. Therefore, we can write:

$$D_o \geq D_{safe} + v_r(T_l + k) + \frac{v_r^2}{2\mu g}$$

from which we derive

$$T_l \leq \frac{D_o - D_{safe}}{v_r} - \frac{v_r}{2\mu g} - k.$$

The same reasoning can be applied to T_v, $TTRT$ to derive the relation between these last parameters and the robots performance, i.e. v_r.

Notice that, if there are fixed obstacles, and D_{max} is the maximum distance a vehicle can detect an object, we can put $D_o = D_{max}$ to obtain the maximum safety velocity, v_{safe}, of the robot from the latter equations. In the presence of mobile obstacles, to compute v_{safe} we set D_o equal to the minimum distance D_{min} at which the robot sensors can detect an object.

We remember that, for achieving a continuous motion, each vehicles' motor must receive a command every 40 msec. The maximum time t_m, that can elapsed between two consecutive commands to each motor is:

$$t_m = T_l + 2TTRT + H_b + T_v + C_{rs} + C_{lc}$$

where C_s, C_r and C_{lc} are the computation times of the tasks in the robot's CPU. When the token is not late, then

$$t_m = T_l + TTRT + T_v + C_{rs} + C_{lc}.$$

5 Experimental results

To verify the correctness of our architecture we implemented a test application in which five robots had to follow a sixth robot, maintaining a predefined formation. For this application we had TTRT = 31.99 msec. Therefore, we chose for the message streams S_b (base station) and S_r (robots) the following parameters $T_b = D_b = T_r = D_r = 30$ msec, which provide a soft guarantee on message delivery. The utilization of the message streams is $U = 0.82$, which is greater than $max(U^*) = 0.34$ (see Section 3.1). The strategy task has a period of 100 msec, the others task have all the same period equal to 40 msec. Relative deadlines are equal to periods. Worst-case computation times are reported in Table 1 and are expressed in milliseconds. Notice that, some of the values of the task set parameters are constrained by the hardware (e.g., the frame grabber acquisition rate) or by feasibility bounds.

Task	τ_l	τ_s	τ_c	τ_r	τ_t	τ_{rr}	τ_{lc}	τ_{rs}	τ_{rt}
C_i	6.5	0.1	0.1	9.5	7	7	6	17.6	1.6

Table 1. Task set parameters.

The maximum time elapsed between two consecutive commands at each robot resulted to be $t_m = 178.32$ msec, and $t_m = 135.65$ msec when the token is not late. From T_l and the others task parameters (see Table 1), we derived $T_{samp} = 144.89$ msec, $T_{react} = 126.73$ msec, and $k_l = 227.67$ msec. Therefore, to perform obstacle avoidance, we set $D_{safe} = 0.5$ cm, $D_o = D_{min} = 3$ cm, so v_{safe} resulted to be 9.26 cm/sec. Assuming the presence of fixed obstacles, the maximum safety velocity is 4.6 m/sec, which is greater than the maximum speed (0.16 m/sec) of the vehicles. This implies that the robots can safely move at their maximum speed. Figure 5 shows the maximum safety velocity v_{safe} of each robot as a function of the minimum distance D_o from an obstacle that guarantees obstacle avoidance, when $T_l = 40$ msec.

Figure 6 reports the maximum value of the strategy task period T_l that guarantees the safe motion of each robot as a function of the maximum safety velocity v_{safe}. Notice that if v_{safe} is higher than a critical speed limit (equal to 10 cm/sec for the illustrated example), the period T_l becomes negative, meaning that no solution can be found to guarantee a safe motion of the robots.

Finally, Figure 7 shows the relation between T_l and the minimum obstacle distance D_o, when v_{safe} is fixed. The dashed line refers to $v_{safe} = 9.26$ cm/sec, which is the maximum safety velocity guaranteed when $T_l = 40$ msec and $D_o = 3$ cm (see above). The solid line refers to $v_{safe} = 16$ cm/sec, which is the maximum speed of the vehicles in our system. Notice that, to improve v_{safe} we must decrease T_l (and/or T_v, $TTRT$), which means upgrading the hardware of the system.

By setting $T_s = 100$ msec and $T_l = 40$ msec, the schedulability of the tasks in the base station can be verified using equation (1).

6 Conclusions

This paper presented an architecture to coordinate a team of autonomous vehicles, which have to interact to accomplish a common goal. The architecture is composed by n nodes: a base station (a personal computer) and $n-1$ mobile robots. The work has been focused on the real-time execution of the tasks running in the various computational components of the system, and on the guarantee of message delivery. We addressed the scheduling problems of the tasks in the base station and presented our solutions on the Shark real-time kernel. Real-time message

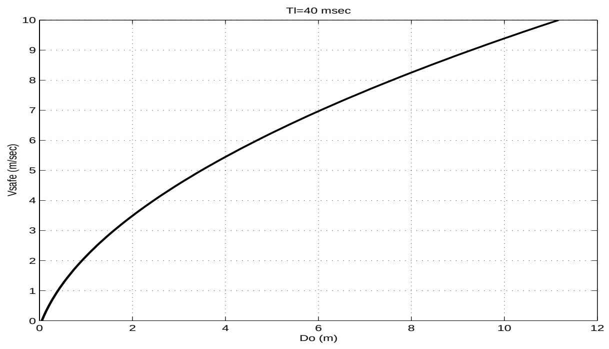

Figure 5. Relation between v_{safe} and obstacle distance D_o with T_l fixed.

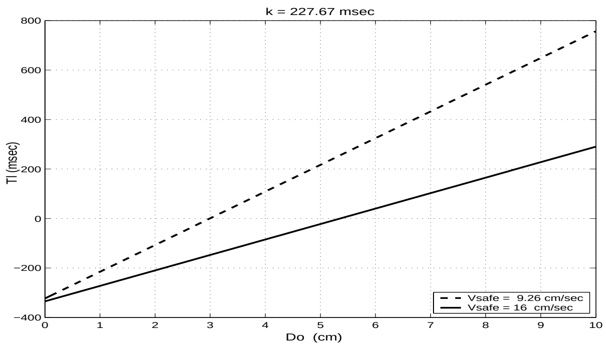

Figure 7. Relation between T_l and D_o with two different v_{safe}.

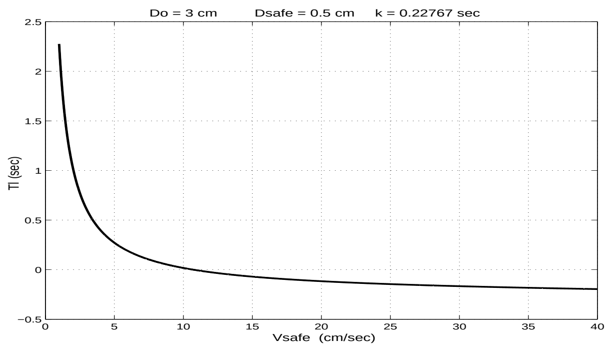

Figure 6. Relation between T_l and v_{safe}.

exchange was obtained by means of a timed-token protocol, that has been analyzed to show how to guarantee message deadlines.

As a future work, we plan to enhance the computational power of the wireless boards in order to test different communication protocols based on EDF message scheduling, to increase the transmission load and coordinate more vehicles. We also plan to replace the CPU on each robot with a more powerful microprocessor, in order to use a real-time kernel also inside the vehicles. Finally, we want to focus our attention on the coordination techniques to better control the robotic team.

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 Computer*, 43(3):327–339, March 1994.

[2] T. P. Baker. Stack-based scheduling of real-time processes. *Real-Time Systems*, 3(1):67–100, March 1991.

[3] G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications - Second Edition.* Springer, 2005.

[4] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *Proc. of the IEEE Real-Time Systems Symposium*, Dec. 2002.

[5] B. Chen and W. Zhao. Properties of the timed token protocol, 92-038. Technical report, Dept. of Computer Science, Texas A&M University College Station, Oct. 1992.

[6] T. Facchinetti, L. Almeida, G. Buttazzo, and C. Marchini. Real-time resource reservation protocol for wireless mobile ad hoc networks. In *Proc. of the IEEE Real-Time Systems Symposium*, Dec. 2004.

[7] T. Facchinetti, G. Buttazzo, M. Caccamo, and L. Almeida. Wireless real-time communication protocol for cooperating mobile units. In *Proc. of the 2nd International Workshop on Real-Time LANs in the Internet Age (RTLIA 2003)*, July 2003.

[8] P. Gai, L. Abeni, M. Giorgi, and G. Buttazzo. A new kernel approach for modular real-time system development. In *Proc. 13th IEEE Euromicro Conf. on Real-Time System*, June 2001.

[9] S. Goddard and X. Liu. A variable rate execution model, tr-unl-cse-2003-15. Technical report, Dept. of Computer Science and Engineering, University of Nebraska - Lincoln, Dec. 2003.

[10] R. Grow. A timed token protocol for local area networks. In *Proc. Electro'82, Token Access Protocols, Paper 17/3*, May 1982.

[11] H. Li, J. Sweeney, K. Ramamritham, R. Grupen, and P. Shenoy. Real-time support for mobile robotics. In *Proc. of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2003)*, May 2003.

[12] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, Feb. 2004.

[13] N. Malcolm and W. Zhao. The timed-token protocol for real-time communications. *Computer*, 27(1):35–41, Jan. 1994.

[14] A. Qadi, S. Goddard, J. Huang, and S. Farritor. A performance and schedulability analysis of an autonomous mobile robot, tr-unl-cse-2004-0015. Technical report, Dept. of Computer Science and Engineering, University of Nebraska - Lincoln, Dec. 2004.

[15] K. Sevcik and M. Johnson. Cycle time properties of the fddi token ring protocol. *IEEE Trans. on Software Eng.*, 13(3):376–385, March 1987.

[16] J. Shi, S. Goddard, A. Lal, and S. Farritor. A real-time model for the robotic highway safety marker system. In *10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004): 331-340*, May 2004.

[17] V. Silva, F. Santos, and L. Almeida. A robust self-localization system for a small mobile autonomous robot. In *Proc. of the 3rd ANIRob/IEEE Int. Symposium on Robotics and Automation*, Sept. 2002.

[18] J. Wu and I. Stojmenovic. Ad-hoc networks. *IEEE Computer*, 37(2), Feb. 2004.

[19] S. Zhang and A. Burns. A study of timing properties with the timed token protocol, ycs 226, 1994. Technical report, University of York, 1994.