Real Time Communications over 802.11: RT-WMP

Danilo Tardioli and José Luis Villarroel
Grupo de Robótica, Percepción y Tiempo Real
Instituto de Investigación en Ingeniería de Aragón
Universidad de Zaragoza
Zaragoza, Spain
Email:{dantard, jlvilla}@unizar.es

Abstract

Ad-hoc networks usually support best-effort traffic and occasionally some kind of Quality of Service (QoS). However, there are some applications, which generally involve cooperative control, with hard real-time traffic requirements where strict deadlines must be met. To meet deadlines, the communication network has to support the timely delivery of inter-task messages. This is the case, for example, of applications involving cooperative robot teams, such as those used for rescue tasks in hostile environments, emergencies or disaster recovery, where a wired backbone is infeasible or economically unviable. In this paper, we present RT-WMP, a novel protocol that allows wireless real-time traffic in relatively small mobile ad-hoc networks using the low-cost commercial 802.11 technology. The protocol is based on a token-passing approach and message exchange is priority based. Moreover, support for frequent topology changes is provided through the sharing of a matrix that describes link quality amongst the members of the network.

1. Introduction

Mobile Ad-hoc Networks (MANETs) are commonly used when the deployment of a wired backbone is infeasible or economically disadvantageous. MANETs usually support best-effort traffic and, in some applications, such as multimedia, can offer some kind of Quality of Service (QoS) like, for example, minimum bandwidth guarantee. However, some applications, generally involving cooperative control, have Hard Real-Time (HRT) traffic requirements. This is the case, for example, of applications involving *cooperative robot teams* (spatial explorations, access to disaster areas where direct human intervention is not suitable, etc.). In these applications, the tasks need to intercommunicate to accomplish a common goal (distributed perception could be one example) and have strict deadlines that must be met. Consequently, the underlying communication network has to support the timely delivery of inter-task messages. Moreover, and in order to meet deadlines, messages have to be properly scheduled for transmission [9] and, therefore, all phases of communication must be time bounded. On the other hand, support for scheduling strategies, such as those based on fixed or dynamic global priorities, allows analysis of HRT distributed systems. However, protocols for Ad-Hoc networks are typically focused on issues such as maximizing throughput or minimizing average message delay, neglecting the indeterminism introduced. On the other hand, most commercial low-level network protocols (e.g. 802.11, 802.15.4, etc.) do not provide timeliness guarantees on network transmissions due to packet collisions, exponential back-offs and the false blocking problem [13]. Nevertheless, to guarantee time boundedness in an HRT communication, access to the medium has to take place in a deterministic manner, as in TDMA or token-based access strategies. Unfortunately, current HRT protocol proposals for MANETs do not offer, at the same time, support for all of the issues involved, such as multi-hop or scheduling strategies and most of them require special hardware as well.

This paper presents the RT-WMP, a novel protocol for *MANETS* that supports *hard real-time* traffic. In fact, in RT-WMP end-to-end message delay has a *bounded and known duration* and it manages global static message *priorities* as well. Besides, RT-WMP supports *multi-hop* communications to increase network coverage. The protocol has been designed to connect a relatively small group (10-20 units maximum) of mobile nodes. It is based on a token passing scheme and is completely decentralized. Any

This work has been funded by the projects NERO DPI2006-07928 (MCYT, Spanish Gov.) and URUS IST-1-045062-URUS-STP (E.C.) and by the agreement between the Gobierno de Aragon and the Universidad de Zaragoza regarding the WALQA research laboratory.

topology of the network will do. The protocol is designed to manage rapid topology changes through the exchange of a new type of adjacency matrix containing link quality amongst nodes. RT-WMP has a built-in efficient error recovery mechanism that can recover from certain types of errors without jeopardizing real-time behavior. A technique for reincorporating lost nodes is proposed as well. The RT-WMP can run over *commercial hardware* without modifications. It is currently *implemented* on the Linux platform and uses commercial low-cost 802.11-protocol [5] based devices. Its functionality and performance have been proven using a robotic team.

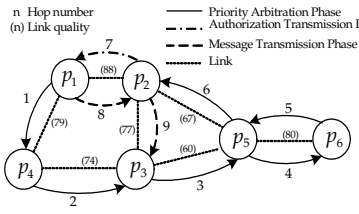
The paper is organized as follows. In section 2 we present related work. In section 3 we present the RT-WMP protocol. In section 4 we discuss the solution adopted by RT-WMP to handle error situations. In section 5 real-time characteristics of RT-WMP are analyzed and section 6 shows experimental results and performance of the proposed protocol, comparing it with the plain 802.11. The article ends with our conclusions and lines of future research.

2. Related works

The literature on how to support real-time communication in wireless environments is not very vast. However, several proposals have been put forward in the last few years. In [19, 12], one or more Access Points coordinate access to the medium. In general, these solutions are improvements on the 802.11-native Point Coordination Function (PCF) protocol, which is infrastructure-based and presents the same restrictions in terms of mobility. In other solutions, a node that needs to transmit occupies the medium with energy pulses [18, 16], the duration of which is proportional to the priority of the node. If nodes try to transmit simultaneously, the station with the highest priority will be the only one to find the medium idle when it ceases to transmit. In this way, the station with the highest priority knows that it won contention for the channel. In general, these solutions do not address the problem of the hidden terminal and require hardware modification. In the WTRP protocol [8], Lee et al. proposed a token-ring network based on the ideas of 802.4 token bus protocol. When a node receives the token, it can transmit for a fixed time. At the end of the transmission, the node passes the token to its successor. Network activity after the token is passed on is interpreted by the sender as an implicit acknowledgment. If acknowledgment fails, the node tries to reconstruct the ring excluding as few nodes as possible. However, in some cases the need to close the ring can lead to the exclusion of many nodes. Besides, multi-hop communication is not supported, since a node can only communicate with its neighbors. Donatiello and Furini [3] propose a similar token-passing solution, in which nodes shall also be organized in a ring. The token always travels in the same direction and messages travel through the nodes belonging to the ring to reach the destination. The need to keep the ring connected introduces many limitations in terms of mobility, since multi-hop communication is possible only by maintaining the ring topology. The solution proposes an interesting spatial-reuse technique based on a Code Division Multiple Access (CDMA) modulation. Unfortunately, CDMA devices are not common consumer products like 802.11 cards, even though this modulation is widely used in mobile phones. Al-Karaki and Chang [1] proposed the EPCF protocol, which is a 802.11-native PCF protocol extension. The enhancement of the protocol is in the polling phase because EPCF incorporates priorities. In the case of multi-hop network environments, some nodes play the role of Virtual Access Point and the net is organized hierarchically. However, the paper does not clearly explain either the steps required to set-up a multi-hop network or the related temporization. Moreover, at the moment there is no existing implementation of this protocol. Recently, in [4], Buttazzo et al. proposed another interesting solution based on a time division scheme. This paper proposes the use of implicit EDF to provide real-time guarantees. Collisions are avoided by replicating and executing the EDF scheduler in parallel in all nodes. Communication amongst nodes is organized in consecutive slots, referred to as system ticks, the duration of which is constant. *Connectivity tracking* is carried out through the exchange of each nodes adjacency matrix, in order to make all the matrices converge toward the unique and correct view of the entire network. However, this solution does not supports user-message multi-hop. In late 2005, though, IEEE approved the 802.11e [6] specification. This standard is a set of technologies for prioritizing traffic, which adds QoS capability to the 802.11 legacy protocol. The 802.11e introduces a new Hybrid Coordination Function (HCF) that replaces the legacy DCF and PCF. Within the HCF, there are two access mechanisms, the enhanced distributed channel access (EDCA) and the HCF controlled channel access (HCCA). While HCCA is a centralized access method, ECDA can be used in ad-hoc networks. EDCA contention access includes priorities by introducing eight priority queues in which messages contend for the right to transmit. However, even if contention windows and backoff times are adjusted to favor messages with the highest priority, collisions can still occur and the resolution mechanism is based on the calculation of a *random* backoff time that is incompatible with real-time planning, just like in the 802.11 legacy protocol. Besides, the legacy 802.11e standard does not offer multi-hop routing and additional routing protocols have to be used. An interesting extension to the 802.11e that includes multi-hop traffic support is presented in [14]. In this solution packets are prioritized using a combination of the laxity of the packet and the number of hops to the destination node to give higher priority to the packets that have

to traverse many hops. However, this solution suppose the modification of the 802.11e protocol to store additional information in its queues. Moreover, like the 802.11e legacy protocol, it has been tought to deal with multimedia traffic that have slightly different requisites from the real-time one.

In short, even if solutions for support of real-time traffic over ad-hoc wireless network exist, there are no solutions that deal globally and completely with real-time and mobile multi-hop requisites. Instead, in some protocols priorities are only supported at node level and not at message level. Besides, very few of the protocols set forth have actually been implemented.



Link Quality Matrix

	p_1	p_2	p_3	p_4	p_5	p_6
p_1	-	88	0	79	0	0
p_2	88	-	77	0	67	0
p_3	0	77	-	74	60	0
p_4	79	0	74	-	0	0
p_5	0	67	60	0	-	80
p_6	0	0	0	0	80	-

Figure 1. A hypothetical situation described by the network graph and the corresponding LQM. The hops sequence of the protocol is also shown.

3. Real-Time Wireless Multi-hop Protocol

RT-WMP is based on the ideas proposed in the Real Time Ethernet Protocol (RT-EP) [10], which introduces the priority based token-passing scheme. RT-WMP completes and extends RT-EP to multi-hop mobile wireless environments with the introduction of a particular type of distance matrix based on link quality amongst nodes.

3.1. Overview

The system architecture considered in this paper consists of a S set of n mobile nodes $S = \{p_0, ..., p_{n-1}\}$ which can communicate over a wireless link. All the nodes use a single *shared* radio channel to exchange messages. We call the subset of nodes that can hear the transmission of node p_i *neighbors* of node p_i. Each node has a transmission and a reception priority queue. Each message exchanged between nodes is identified by a priority level in the 0..127 range, where 127 is the highest priority value. Messages with the same priority are stored in FIFO order. When an application needs to transmit a message to another node, it pushes it into the transmission queue. The RT-WMP process pops the message from that queue and transmits it through the network to the destination node. The latter pushes the message into the reception queue and the destination application can finally pop the message from that queue.

Protocol operations. The protocol works in three phases (see Fig.1): *Priority Arbitration phase (PAP)*, *Authorization Transmission Phase (ATP)* and *Message Transmission Phase (MTP)*. During the PAP, nodes reach a consensus over which of them holds the Most Priority Message (MPM) in the network in that moment. Subsequently, in the ATP, an authorization to transmit is sent to the node which holds the highest priority message. Finally, in the MTP, this node sends the message to the destination node.

To reach a consensus over which node holds the highest priority message, in the PAP a token travels through all of the nodes. The token holds information on the priority level of the MPM in the network and its owner amongst the set of nodes already reached by the token. The node which initiates the PAP states that the highest priority message in its own queue is the MPM in the whole network and stores this information in the token. Then it sends the token to another node, which checks the messages in its own queue. If the node verifies that it holds a message with a higher priority level than the one carried by the token, it modifies the token data and continues the phase. The last node to receive the token, which knows the identity of the MPM holder, closes the PAP and initiates the ATP. In this phase, the node calculates a path to the MPM holder using the topology information shared amongst the members of the network (the Link Quality Matrix, see below) and sends an authorization message to the first node in the path. The latter will route the message to the second node in the path and so on, until the authorization reaches the MPM holder. This is when the MTP begins. The development of this phase is quite similar to the preceding one. The node that has received the authorization calculates the path to reach the destination, and sends the message to the first node of the path. The message follows the path and eventually reaches its destination. Phases repeat one after another i.e., when the MTP finishes, the node destination of the message initiates a new PAP an so on. When none of the nodes have a message to transmit, the authorization and message transmission phase are omitted and priority arbitration phases repeat continuously.

3.2. The Link Quality Matrix

To describe the topology of the network, RT-WMP defines an extension of the network connectivity graph (as defined in [4]) adding nonnegative values on the edges of the graph. These values are calculated as functions of the *radio signal* between pairs of nodes and are indicators of *link quality* between them. These values are represented in a matrix which we call Link Quality Matrix (LQM), the elements of which $lqm_{ij} \epsilon [0, max\_lq]$ describe link quality between p_i and p_j nodes (see Fig.1). Each column LQM_k describes the links of the p_k node with its neighbors. Even if the links

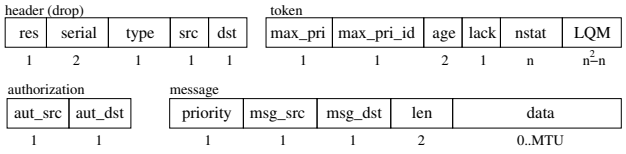


Figure 2. Frames of the protocol. Fields size is expressed in bytes.

can be asymmetric (the radio signal received by p_i when p_j transmits can be different from the one received by p_j when $p_i$transmits), generally the differences are small and in this paper we presuppose them to be symmetrical. Consequently, the resulting LQM is symmetrical as well. Nodes use this matrix to select which node to pass the token to and to take decisions on the best path to route a message from a source to a destination. All the nodes have a local copy of the LQM that is updated each time a frame is received. Besides, every node is responsible for updating its column of the LQM (both in the local copy and the shared copy) to inform the other nodes about local topology changes.

3.3. Frames definition

In Fig.2 we present the frames exchanged amongst nodes. Frames have a common header and an extra part that is different for each frame type. In the header, the first byte (*res*) is reserved for communication between the network interface card (NIC) driver and the RT-WMP process. The *serial* field contains the serial number of the frame. This field is used in the error recovery mechanism (see section 4). The *type* field identifies the type of the frame (*token, authorization, message* or *drop*). The *src* and *dst* fields contain information about the source and the destination of the frame. In fact, in the RT-WMP, nodes are identified through a natural number between 0 and $n-1$ called WMP address, n being the fixed number of the nodes in the network. When a node needs to send a frame (of any type), it fills the *src* and *dst* fields of the header with its WMP address and the WMP address of the destination node and broadcasts the frame. Since the radio channel is shared, all the neighbors of the sender hear the frame but only the destination processes it. The *token* frame adds the *max_pri* and *max_pri_id* fields that carry the MPM priority level and the MPM holder WMP address. The age field is used to find the oldest message amongst messages with the same priority level. The *lack* field is used for *belated acknowledgement* of the sender (see section 3.4). The *nstat* field is an array of *n* bytes. The value *nstat[i]* represents the status of the p_i node that can be *unreached, reached, lost* and *searched*. Finally, the *LQM* field contains the Link Quality Matrix. The *authorization* frame adds the *aut_dst* and *aut_src* fields that carry the address of the destination node and of the source of the authorization to the common header. The *message* frame type holds the *msg_src* and *msg_dst* field that contain the WMP address of the source and of the messages destination. The *priority* and *len* fields hold the priority and the length of the data carried by the frame as well. The field *data* contains the payload of the frame, which can have a length between 0 and MTU bytes. Finally, the *drop* frame is a simple header and is identified through the *type* field.

3.4. Phases of the protocol

In the following sections, we offer a detailed description of the three phases of the protocol. Let us suppose that all the nodes know the network topology (i.e. all the nodes have the same LQM) and that the network is connected. In these sections, we also presuppose that the nodes stay put and that communication is error free. These limitations will be treated in the sections 3.5 and 4 respectively.

Priority Arbitration Phase. The first phase is the priority arbitration phase. When a p_k node initiates the PAP, it creates a new token, copies its local LQM in the relevant field of the token and sets the $nstat[i]$ to *unreached* $\forall\, i \in [0, n-1] \,:\, i \neq k$. The value $nstat[k]$ will be set instead to *reached*. This means that, in the current PAP, none of the nodes have been reached by the token yet, except the p_k node. Afterward, it checks the priority level of the highest priority message in its transmission queue and sets the *max_pri* and *max_pri_id* fields with this value and its WMP address respectively. The age field is filled with the age of the message (i.e. the time that the message has spent in the queue up to that moment) expressed in milliseconds. In this way, the p_k node is stating that it is the MPM holder. Then, it analyzes the LQM to know with which p_{bl} node shares the best link quality, and sends the token to it. When p_{bl} receives the token, it sets the $nstat[bl]$ to *reached*, it updates the *LQM* token field with its local data and saves the matrix locally. It subsequently increases the value of the age field by a quantity equal to the duration of one token-pass hop, in order to update the age of the message that the token refers to. Then it looks for the value of the *max_pri* field of the token and compares it with the priority level of the highest priority message in its queue. If it verifies that it holds a higher priority message, it modifies the *max_pri* and *max_pri_id* fields. If it holds a message with the same priority, however, it checks the age field of the token. If the message is older than the one carried by the token, it updates the token as well. Subsequently, it chooses the node with which it shares the best link quality *amongst the set of nodes not yet reached*, and sends the token to it. If a node only listens to its predecessor (i.e. the node that passed the token to it), it can return the token to the predecessor af-

ter updating. This means that a node can receive the token several times during the same PAP. In that case it has the right to update the *max_pri* and *max_pri_id* values. This behavior helps reduce the well-known priority inversion problem. The process is repeated until all the nodes have been reached by the token (i.e. $nstat[i] = reached \, \forall \, i$). The last node to receive the token knows the MPM holder's identity (which is contained in the *max_pri_id* field) and is responsible for sending it the authorization. This node ends the PAP and initiates the ATP.

Authorization Transmission Phase. First of all, the node that starts the ATP calculates a path to reach the destination node. To do this, it applies the well known Dijkstra algorithm [2] for single-source shortest path problem to a distance matrix derived from the LQM as described in section 3.5. The Dijkstra algorithm returns a path to the destination as a set $P = \{p_{p_1}..p_{p_m}\}$ of nodes. Then the node creates an *authorization* and fills the *aut_dest* and *aut_src* fields with the MPM holder address and its own address respectively, and sends the authorization to the first node of the path. When p_{p_1} receives the authorization, it looks at the *aut_dest* field and if it contains its address, it ends the ATP and initiates the MTP. Otherwise, it calculates the $P' = \{p'_{p_1}..p'_{p_{(m-1)}}\}$ path, where $p'_{p_k} = p_{p_{(k+1)} \; k<m}$. In other words, since the calculation is executed over the same LQM, the path calculated will be the same, except that the first hop has already taken place. Since all the nodes have the same topological information, recalculation of the path in each hop allows the saving of bandwidth needed to propagate it. The node repeats the process just explained, routing the message to the next member of the path, leaving the *aut_dest* field unchanged.

Message Transmission Phase. When the MPM holder receives the authorization to transmit, it takes the highest priority message out from its transmission queue, creates a new *message* frame and places the data in the *data* field. It fills the *msg_src* and *msg_dest* fields with its address and with the destination address and calculates the path to the destination, just like in the ATP. Then it fills the *priority* and *len* fields with the messages priority and data length and sends it to the first node that belongs to the path. When the latter receives the message, it checks the *msg_dest* field. If it contains its address (i.e. if it is the destination) it pushes the message into the receipt queue and starts a new PAP. Otherwise, it repeats the computation of the path and repeats the process just explained, routing the message to the next member of the path and leaving the *msg_dest* field unchanged. An explicit acknowledgment is not included because it would create too much overhead. However, if the message reaches the destination node, the latter introduces its WMP address in the *lack* field of the new token before initiating the new PAP. During this PAP, the token will reach the sender of the previous message, who can check if the message has been delivered or not by looking at the *lack* field.

3.5. Mobility management

Topology can change frequently in MANETS. If nodes are moving, the radio signal and therefore link quality amongst them varies and these changes must be rapidly reflected in the global status of the network. Consequently, when a node discovers a change, it has to propagate this information as soon as possible. In RT-WMP this task is carried out by the token in the PAP phases. In fact, topological information travels with it in the LQM and is updated in each hop.

LQM actualization. As explained earlier, each column LQM_k of the LQM describes the links of the p_k node with its neighbors. Nodes can easily obtain information to fill the relevant column. In fact, when a node sends a frame of any type, its neighbors due to the broadcasting nature of the wireless medium listen to the transmission and read the radio signal from the network layer to update its local LQM. These changes have to be reflected in the shared LQM as soon as possible, to allow the nodes to correctly calculate the paths in the ATP and MTP. Therefore, when a p_k node receives a token, it updates the LQM_k column of the *LQM* token field, saves the whole matrix locally to use it in the successive ATP and MTP, and then retransmits it. Since the LQM reaches all the nodes frequently, they have accurate and up-to-date information on the network's (link-quality) topology. Consequently, if two nodes are moving away from each other, link quality will gradually fall until it reaches a value close to zero, after which the link is lost. This value is reflected in the LQM, and nodes, whenever possible, will avoid that link to route information when link quality is beneath a certain threshold (see below). Anyway, due to the technique used to update the LQM, the value 0 never would appear, and the last positive value would be maintained until an error were verified (see section 4). To avoid this behavior, we introduced a timeout on the validity of the values of the local LQM elements. If the p_k node has a local LQM where $lqm_{kl} > 0$ but does not hear a transmission from node p_l within a certain timeout, then p_k supposes that p_l is not near it yet, and sets that value to 0. In the worst of cases, the frequency with which each node receives the token depends on the number of nodes, on the maximum message size and on network rate. However, in a 11 Mbps 802.11 network with 10 nodes and messages 1500 bytes long, if a node is moving at 5m/s (18Km/h), it may have moved about 7.5 cm between the reception of a token and the next one (including an ATP and an MTP). This

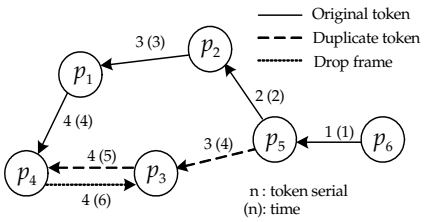


Figure 3. Token duplication resolution mechanism. In case of message or authorization duplication the mechanism works in a similar way.

guarantees that LQM reflects the topology of the network well at all times. However, there is an additional method to maintain the LQM up-to-date. When a node sends a token, all of its neighbors receive the frame. While the destination processes the frame and makes the appropriate decision, the other nodes update their local information on link quality with the sender, as explained earlier. Moreover, the token received contains a more recent LQM that nodes can use to update their own.

When the protocol begins, nodes do not have topological information. Therefore, an additional step to setup the initial LQM is required. This is an easy and bounded time process in which, however, real-time behavior is not guaranteed, and which is not explained here due to lack of space.

Specific LQM elements values. As anticipated earlier, the lqm_{ij} elements of the LQM are functions of the link radio signal between nodes. To calculate them, we use the *Received Signal Strength Indicator* (RSSI) defined by the 802.11 protocol. The physical sublayer measures the energy observed at the antenna used to receive the current frame. Normally, 802.11 devices provide this value to the device driver. Besides, some card models provide information on noise as well. With these two parameters, we can estimate the Signal to Noise Ratio (SNR) for every frame received and estimate link quality between nodes, representing it with values in the $[0, max\_lq]$ range. The calculation of the path in the ATP and MTP is based on these values. The Dijkstra algorithm for single-source shortest path problem for directed graphs with nonnegative edge weights is applied to the graph that represents the matrix M, derived by means of a simple heuristic from the LQM. However, the heuristic is not linear, as longer paths are preferred to shorter ones containing very bad links. Computation time is not an issue, since the Dijkstra algorithm has a $O(V^2)$ complexity, V being the number of vertices, whereas the implementation of the algorithm's priority queue with a Fibonacci heap makes the time complexity $O(E + VlogV)$, where E is the number of edges of the considered graph.

4. Error handling in RT-WMP

There are basically two possible causes of error that have to be managed in this protocol: node failure and communication error. While the former is common to any network system, the latter is specially important in wireless networks. The error recovery mechanisms of RT-WMP have been designed to not jeopardize real-time behavior of the protocol and to maintain the network temporization in the majority of possible situations of error.

4.1. Node failure or node loss

RT-WMP is quite robust in case of node failure. The *implicit acknowledge* technique used dispenses with the necessity of monitor nodes to control the loss of the token. In fact, in common token-pass systems with explicit acknowledgement, when a node receives a token, it acknowledges the sender with a message. However, if a node falls just after the acknowledgement, the token is lost and a technique to regenerate it is required. In RT-WMP, instead, when a p_k node sends a frame of any type, it listens to the channel for a timeout. The receiver p_l node immediately processes the frame received and sends another frame to a third p_m node (that can be its predecessor as well). The first sender listens to such a frame as well and interprets it as an acknowledgement. This technique permits saving of bandwidth and eliminates the need for a monitor node. Anyway, if the first sender does not hear the frame within timeout, it supposes that the p_l node has fallen or is out of its coverage range. In this case, behavior depends on the phase that the protocol is in. If it is in the ATP or MTP, p_k discards the frame and starts a new PAP. In fact, it is impossible to calculate another path since this could jeopardize the temporization of the network (see section 5.1). However, if it is in the PAP, p_k node sets the $nstat[l]$ field to *reached*, modifies the local LQM and the LQM carried by the token to exclude p_l node from the set of its neighbors (setting $lqm_{kl} = 0$) and continues with the PAP, sending the token to other node. This solution excludes the p_l node in the current PAP to preserve network temporization but not necessarily in the next. In fact, it may be that there are other nodes which consider p_l its neighbor. If p_l has not actually fallen but, for instance, has moved away from p_k node but not from another neighbor p_m, the latter will reinsert p_l in the next PAP by simply passing it the token with no additional cost. If, instead, the node is actually broken or has moved away from all the other nodes, in the next PAPs all its neighbors will try to pass the token to it one after another (one in each PAP) until p_l is isolated. When this occurs, the node that starts the next PAP marks this node as *lost* setting $nstat(l) = lost + r$ where r is a number between 0 and $n - 1$ and p_r does not belong to the set of lost nodes.

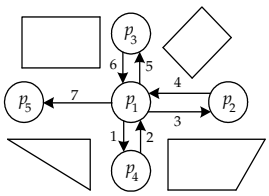


Figure 4. Worst case PAP situation.

Reinsertion of lost nodes. This number represents the identity of the node that has to *search* for the lost node in the current PAP. Nodes, in fact, do reappear, but it is impossible to predict where. Consequently, nodes that still belong to the network organize themselves to search for the lost node one after another in the successive PAPs. When p_r node receives the token, it looks at the $nstat$ array. If one of the elements contains the value $lost + r$, (in this case $nstat[l] = lost+r$), it tries to send the token to p_l. If the latter (implicitly) acknowledges the frame (i.e. passes the token to another node or back to p_r), it is reinserted in the network with no additional cost. Otherwise, (i.e. p_l node does not acknowledge) node p_r sets $nstats(l) = searched + r$ and continues the PAP. None of the other nodes try to search for that node in the current PAP, since this would break the network temporization (see 5.1 for details). The node that starts the next PAP modifies the field $nstats$ to $nstats[l] = lost + ((r + 1)\ mod\ n)$ if $p_{(r+1)\ mod\ n}$ node is not a lost node and continues the PAP. In this manner all the nodes not lost will search for the lost node one after another in the successive PAPs until reinsertion of the node takes place. This mechanism works with more than one lost node in the same way.

4.2. Frame duplication

Communication errors can produce another type of problem. Let us consider the situation where, in the PAP, the p_k node sends a token to the p_a node and waits for an implicit acknowledgment. Node p_a processes the frame and sends the frame to node p_b. As explained earlier, the last pass is also the acknowledgement for p_k . However, if node p_b hears the frame but p_k does not, a token duplication occurs. In fact, p_k marks the node as *reached* (like in the case of a failed token-pass explained earlier) and continues the PAP by sending the token to another node. Node p_b continues the PAP as well and at that moment there are two tokens in the network. To solve this problem we introduced the *serial* field in the frames. Before each transmission, the sender node increments this value, saves it locally, and then transmits. However, when a node receives a frame with a *serial* lower or equal than the *highest* serial that has transmitted, it discards the frame and informs the sender by sending a *drop* frame to it. In Fig.3 an example situation is presented.

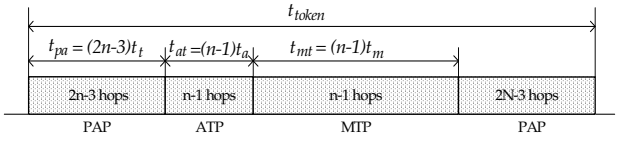


Figure 5. Timing of the protocol.

Authorization or *message* duplication can occur in the same way. In keeping with the behavior described earlier, the unacknowledged node discards the *message* or the *authorization* and creates a new *token* frame. However, the receiver of the authorization/message continues to route the frame along the path. At that moment there are two distinct types of frame traveling in the network. Just as in the case of a simple token duplication, though, the first node that receives a frame with an old *serial* (either token or authorization/message) will discard it.

5. Real-time features

In a real-time distributed system, priority support and bounded end-to-end delay is required for scheduling and time constraint guarantees. Consequently, each event/phase in a real-time network protocol *must* have a *bounded* and *known* duration.

5.1. RT-WMP phases boundness

The RT-WMP fulfils these requirements, since each phase has a bounded and known duration, even in the presence of the majority of errors. The PAP lasts, in the worst-case, $2n - 3$ hops (see Fig.4). In fact, if the network is connected, a covering tree with $n - 1$ arcs can always be found, so the tree can be covered by visiting all its nodes two times at the most. That would mean $2n - 2$ hops, but a return to the first node can be avoided; therefore, there are only $2n - 3$ hops. Besides, in error situations, such as node failure or node reinsertion (or loss), this phase has *the same* duration, since the time wasted on a failed token pass can be equated to the time needed to send and return a token. Let us consider an example. Suppose that p_k is the only neighbor of p_a. In the normal functioning of the protocol, if node p_k sends the token to p_a, the latter shall pass the token back to p_k to continue the PAP. However if node p_a is broken, after the pass, node p_k will wait for a timeout and then continues with the PAP, passing the token on to another node. Actually, the time spent in the first and second scenario is the same, since the timeout is equal to the time that node p_a needs to return the token. Token duplication is the only situation that can temporarily jeopardize real-time behavior of the network due to the possibility of collisions and the need to send drop frames. Anyway, the duration of this sit-

Table 1. Timing of the protocol for 11 Mbps data rate and message 512 bytes long. Times are expressed in ms.

Nodes	3	4	5	10	20
t_t	0.282	0.287	0.295	0.353	0.578
t_a	0.271	0.271	0.271	0.271	0.271
t_m	0.646	0.646	0.646	0.646	0.646
t_{pa}	0.846	1.43	2.06	6.02	21.4
t_{at}	0.542	0.813	1.08	2.43	5.15
t_{mt}	1.29	1.93	2.58	5.81	12.28
t_{token}	3.52	5.63	7.80	20.2	60.2
t_{wc}	5.36	8.38	11.4	28.5	77.6

uation is limited since the first node to receive an old frame discards it and informs the sender that its frame is old.

In the ATP and MTP, the path is determined using the well-known Dijkstra algorithm (see section 3.5 for details). According to this algorithm, if the network is connected, the maximum number of hops to go from one node to another is $n-1$. Thus these phases have a bounded duration as well. However, in case of node failure or loss during these phases, nodes shall discard the authorization or the message in order not to jeopardize the temporization of the network. For example, let us suppose that p_k node calculates a $n-1$ *hops* path to reach destination node p_d. If one hop along the path fails, for instance p_a to p_b, p_a node might find another path to reach p_d. Unfortunately, it may also occur that the sum between the number of hops already made and the number of hops of the new path is superior to the allowed maximum, that is $n-1$.

5.2. RT-WMP timing and bandwidth

From the global point of view, the phases of the protocol repeat one after another, as we can see in Fig.5 with worst-case durations $t_{pa} = (2n-3)t_t$ for the PA phase, $t_{at} = (n-1)t_a$ for the ATP and $t_{mt} = (n-1)t_m$ for the MTP, t_t being the duration of a *token* pass, t_a the duration of an *authorization* pass and t_m the duration of a *message* pass. The absolute values of t_t, t_a and t_m depend on protocol parameters such as the number of nodes, the data rate and the maximum transmission unit (MTU) that the network has to carry, as well as on the underlying 802.11 protocol.

Details of the operation of the 802.11 protocol are not the subject of this paper and it suffices to know that before transmitting, a node always has to wait for a time interval called DCF Interframe Space (DIFS). The DIFS has a fixed duration of $50\mu s$, which has to be taken into account when values t_t, t_a and t_m are calculated. Regardless of network speed, however, there is a part of the 802.11 frame that is always sent at 1 Mbps (the PLPC preamble and header) and has a fixed duration of $192\mu s$. Finally, we have the MAC header and the FCS, which together add 28 bytes and are transmitted at the actual effective bitrate.

Timing. Taking into account the parameters just presented, the transmission duration of frames can be calculated as:

$$t_{frame}(\mu s) = (192 + 50) + \frac{(28 + L) \cdot 8}{tx\_rate(bps)}$$

L being the payload *of the 802.11 frame*, including the overhead protocol (i.e. $L = 11 + n^2$ for the token, $L = 8$ for the authorization and $L = 11 + MTU$ for the messages). With these data we can calculate that, in the worst-case, the token reaches each node every:

$$t_{token} = 2t_{pa} + t_{at} + t_{mt}$$

On the other hand, the highest priority message in the network (if it is the only one) could suffer, in the worst-case, a delay of:

$$t_{wc} = 2 \cdot (t_{pa} + t_{at} + t_{mt})$$

Table 1 illustrates a few results for different values of the number of the nodes parameter for a 11 Mbps network rate and a MTU of 512 bytes.

Bandwidth. With the temporization just expounded we can easily calculate the worst-case theoretical (end-to-end) bandwidth offered by the protocol as follows:

$$BW(Mbps) = \frac{MTU \cdot 8}{t_{pa}(\mu s) + t_{at}(\mu s) + t_{mt}(\mu s)}$$

Our goal was to compare it with the 802.11 protocol to offer a rough idea of the overhead imposed by RT-WMP. However, before analyzing comparison results, it would be convenient to raise a few issues concerning 802.11 throughput. According to [7], the theoretical (in absence of errors) effective available bandwidth offered by 802.11b protocol (at 11 Mbps) is about 6.1 Mbps for packets 1500 bytes long. If the RTS/CTS mechanism is used to alleviate the hidden terminal problem, this value drops to about 4.5 Mbps. On the other hand, 802.11 does not support multi-hop routing and this feature has to be implemented by means of upper layer routing protocols such as AODV, DSVD, etc. However, regardless of the overhead introduced by the routing protocol used, end-to-end bandwidth is highly dependent on network topology and the number of nodes in the network. In fact, according to [17], a transmission can cause interference in a range larger than the range of communication (almost two times the latter). Nodes within the *carrier sensing range* of a transmitting node can sense the carrier

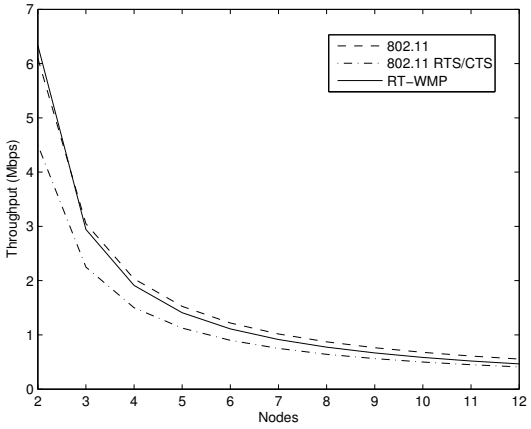


Figure 6. Comparison between RT-WMP and 802.11 for the worst-case situation.

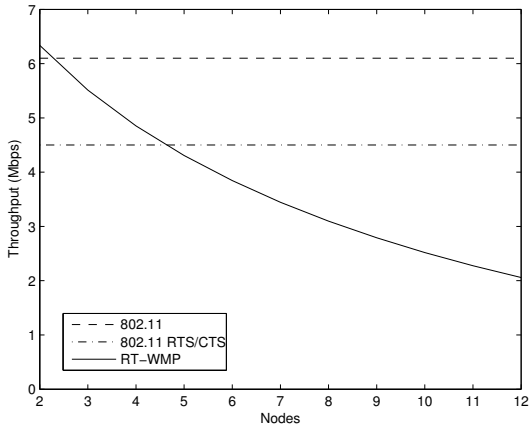


Figure 7. Comparison between RT-WMP and 802.11 for the best-case situation.

of the sender even if they cannot hear the frame, and thus delay its transmission. According to our research, in relatively small wireless networks (up to 10-12 nodes) there can be situations where each node can only communicate with its predecessor and its successor, and carrier sensing does not allow spatial reuse (i.e. only *one* node can transmit at a time). In this case, end-to-end bandwidth depends on the number of hops separating the sender from the receiver. The *worst-case situation* occurs when the source and the destination are $n-1$ hops away, n being the number of nodes in the network. In this case the available bandwidth of the 802.11 protocol can be expressed as:

$$BW_{end\_to\_end} = \frac{BW_{channel}}{(n-1)}$$

In [11] through simulation and experimental results, the authors show that in a four node chain, *end-to-end* throughput can reach values close to 2 Mbps, whereas for a six node chain the result is approximately 1.2 Mbps, values that match our calculus. Figure 6 shows the results of the comparison for an 11 Mbps data rate and 1500 bytes data frames. As we can see, in the situation just described, RT-WMP achieves better results than 802.11 using the RTS/CTS mechanism, and very similar ones to the plain 802.11 protocol. Despite the overhead added by the protocol, the use of broadcast frames permits time saving, thanks to the absence of the acknowledgment frames and the respective Interframe Space (IFS). In Fig.7 the comparison between the protocols in the best situation (i.e. when all the nodes can communicate with each other) is shown. In this case, RT-WMP pays for the time spent in the PA and AT phases. However, the results are similar with 802.11, particularly for network with a small sets of nodes.

6. Performance evaluation

In order to verify the performance of the protocol, we used four Versalogic VSBC-8 PCs equipped with Pentium III at 800MHz and Cisco 350 series IEEE802.11b devices, installed in four Pioneer 3AT Mobile Robots from ActivMedia. We used the Debian Linux operating system with kernel version 2.6.8. The RT-WMP is currently implemented over 802.11, IP and UDP. In the future, the IP and UDP layers will be removed to improve performance. In fact, UDP and IP headers add up to 28 overhead bytes, which are completely useless to RT-WMP.

6.1. Real-time behavior

We were interested in verifying several characteristic of RT-WMP and its correct implementation.

The first one was the priority-based message exchange mechanism. To do this, we performed the following experiment. In a four-node network, saturated traffic was generated in all of the nodes. Nodes had a transmission priority queue of 20 messages and the messages generated had a random priority in a range between 0 and 31. Our goal was to measure the delay suffered by messages according to their priority. As shown in Fig.8, in each node, messages with low priority suffer from larger delays, while for high priority messages delays are very short. In other words, as we expected, message delay is a function of the inverse of the message priority.

The second experiment concerned the fairness of the protocol. In this case all the messages (generated in saturated manner in all the nodes) had equal priority. The goal was to verify that all the nodes had the same chances of sending their messages. In other words, that all messages

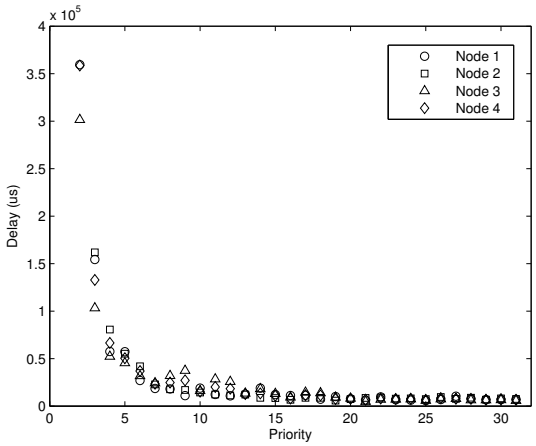


Figure 8. Priority behavior of the protocol.

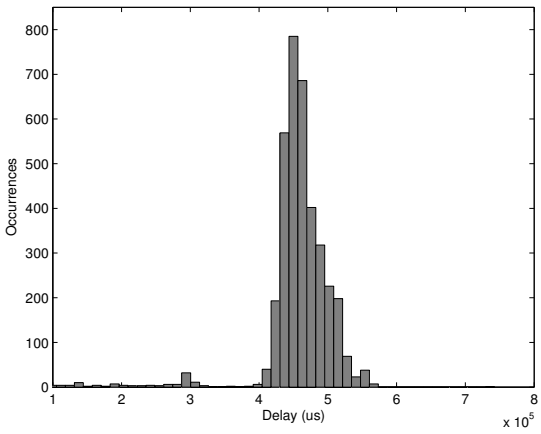


Figure 9. Fairness of the protocol.

experiment the same delay between their entry in the queue and their exit. In Fig.9 the results of this experiment are shown. As can be seen, the implementation fulfils the requirements.

6.2. Throughput

To verify the real end-to-end throughput of the protocol we performed two types of experiment. In both experiments the underlying 802.11 protocol's network rate was 11 Mbps.

For the best-case, we created a completely connected network (placing the robots close one to another) of two, three and four nodes and saturated traffic was generated in all the nodes. In this situation the PAPs always lasted $n-1$ hops, the ATPs zero (when the authorized node is the one that closes the ATP) or one hop and MTPs lasted only one hop. To determine the effective instantaneous bandwidth, we divided the payload of a message by the time lapse measured between the creation of a new token and the delivery of the corresponding message.

To test throughput in the worst-case situation (those where PAPs last $2n-3$, ATPs and MTPs last $n-1$ hops), we performed three tests with n equal to two, three and four nodes. We provided the nodes with a fake LQM to simulate a chain of two, three and four nodes respectively and saturated traffic was generated in all the nodes. With this configuration, worst-case situations can, in fact, occur. We calculated the effective instantaneous bandwidth just as we did in the best-case but this time only taking into account the intervals in which the worst-case situation occurred.

We then averaged the measurement and obtained the results shown in Table 2. The values are quite different to the theoretical ones. However, since the execution time of the protocol code is very small, the bottleneck is constituted by the Linux network layer and the extra UDP/IP layers used. In fact, according to our measurement, from the moment in which the network card driver throws the interrupt to communicate the arrivals of a packet until the moment in which the application layer receives the data, there is a delay of somewhere between 0.5 and 1.5 ms. This delay has a stronger effect when packet size is small, whereas when packet size increases, experimental results get closer to the theoretical ones. To solve this problem, we are currently implementing the RT-WMP in the Linux Kernel Space as Kernel Module and optimizing the network drivers to suit our needs.

7. Conclusions

In this paper, we have presented the basic features of the RT-WMP, which is a new protocol that can work over commercial low-cost 802.11-protocol based networks providing hard real-time traffic support. It uses a token-passing scheme to guarantee bounded transmission times and has message priority support. The protocol deals with frequent topology changes through the sharing of matrices that describes link quality amongst nodes. In addition, we have defined error management and recovery aspects that deal with token loss, token duplication and node reincorporation after a single or multiple failure. These features are implemented while maintaining real-time behavior in the majority of possible error situations. The theoretical analysis shows that RT-WMP offers a bandwidth similar or better than the one offered by the 802.11 plain protocol in worst-case multi-hop situations, and comparable to it in relatively small and completely connected networks.

At the moment the protocol is implemented over the Linux operating system and is being used in real multi-robot application. We are presently working on the development of a formal method of analysis that allows verification of the

Table 2. Real end-to-end bandwidth experiments results (Mbps).

	2 Nodes		3 Nodes		4 Nodes	
Size (Bytes)	Worst Case	Best Case	Worst Case	Best Case	Worst Case	Best Case
100	0.258	0.284	0.177	0.262	0.110	0.247
256	0.830	0.893	0.387	0.573	0.249	0.478
512	1.47	1.62	0.658	1.05	0.422	0.875
1024	2.06	2.19	0. 975	1.54	0.645	1.46
1500	2.28	2.33	1.26	2.27	0.813	1.79

temporal requirements of messages and tasks in distributed systems. We are also rewriting the code to get this protocol to function as Linux Kernel Module in order to improve performance. In the near future, we intend to get this protocol to work over the MarteOS [15] real-time operating system.

Acknowledgment

Special thanks to Michael González Harbour and his team at the University of Cantabria for his precious contribution to this paper.

References

[1] J. N. Al-Karaki and J. M. Chang. Quality of service support in IEEE 802.11 wireless ad hoc networks. *Ad Hoc Networks*, 2(3):265–281, 2004.

[2] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[3] L. Donatiello and M. Furini. Ad Hoc Networks: A Protocol for Supporting QoS Applications. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 219.2, Washington, DC, USA, 2003. IEEE Computer Society.

[4] T. Facchinetti, G. Buttazzo, and L. Almeida. Dynamic resource reservation and connectivity tracking to support real-time communication among mobile units. *EURASIP J. Wirel. Commun. Netw.*, 5(5):712–730, 2005.

[5] IEEE ComputerSociety LAN MAN Stradards Committee. IEEE Std. 802.11-1997, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1997.

[6] IEEE ComputerSociety LAN MAN Stradards Committee. IEEE Std. 802.11e-2005, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, 2005.

[7] J. Jun, P. Peddabachagari, and M. Sichitiu. Theoretical Maximum Throughput of IEEE 802.11 and its Applications. In *NCA '03: Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, page 249, Washington, DC, USA, 2003. IEEE Computer Society.

[8] D. Lee, R. Attias, A. Puri, R. Sengupta, S. Tripakis, and P. Varaiya. A Wireless Token Ring Protocol For Ad-Hoc Networks. In *Aerospace Conference Proceedings, 2002. IEEE*, 2002.

[9] N. Malcolm and W. Zhao. Hard real-time communication in multiple-access networks. *Real-Time Syst.*, 8(1):35–77, 1995.

[10] J. M. Martínez and M. G. Harbour. RT-EP: A Fixed-Priority Real Time Communication Protocol over Standard Ethernet. In *Ada-Europe*, pages 180–195, 2005.

[11] P. Ng and S. Liew. Throughput Analysis of IEEE 802.11 Multi-hop Ad hoc Networks. *to appear in IEEE/ACM Trans. Netw.*, 2007.

[12] P. Pradhan and T. Chiueh. Real-Time Performance Guarantees over Wired/Wireless LANs. In *RTAS '98: Proceedings of the Fourth IEEE Real-Time Technology and Applications Symposium*, page 29, Washington, DC, USA, 1998. IEEE Computer Society.

[13] S. Ray, J. B. Carruthers, and D. Starobinski. RTS/CTS-induced congestion in ad hoc wireless LANs. In *Wireless Communications and Networking, 2003*, volume 3, pages 1516–1521 vol.3, 2003.

[14] T. B. Reddy, J. P. John, and C. S. R. Murthy. Providing MAC QoS for multimedia traffic in 802.11e based multi-hop ad hoc wireless networks. *Comput. Networks*, 51(1):153–176, 2007.

[15] M. A. Rivas and M. G. Harbour. MaRTE OS: An Ada Kernel for Real-Time Embedded Applications. *LNCS*, 2043:305–317.

[16] J. Sheu, C. Liu, S. Wu, and Y. Tseng. A priority MAC protocol to support real-time traffic in ad hoc networks. *Wirel. Netw.*, 10(1):61–69, 2004.

[17] J. Sobrinho and A. Krishnakumar. Quality-of-service in ad hoc carrier sense multiple access wireless networks. *IEEE J. Sel. Areas Commun.*, 17(8):1353–1368, Aug. 1999.

[18] J. L. Sobrinho and A. S. Krishnakumar. Real-time traffic over the IEEE 802.11 medium access control layer. *Bell Labs Technical Journal*, 1(2):172–87, 1996.

[19] H. Ye, G. C. Walsh, and L. Bushnell. Real-Time Mixed Traffic Wireless Networks. *IEEE Trans. Ind. Electron.*, 8(5), Oct. 2001.