RTCAN
A Real-time CAN-bus Protocol for Robotic Applications

Martino Migliavacca, Andrea Bonarini and Matteo Matteucci
Dipartimento di Elettronica, Informazione e Bioingegneria - DEIB, Politecnico di Milano, Milano, Italy

Keywords: Real-time Communication, CAN-bus, Robot Design, Distributed Architecture for Robots.

Abstract: Robots are distributed systems where different devices perform specific tasks and need to exchange data to run the overall system. In this paper, the communication requirements of robotic systems are summarized, highlighting which characteristics are relevant to the different tasks and showing the limits of the present communication protocols. Then, RTCAN is presented: a new real-time CAN-Bus protocol for robotic applications, which aims at combining the advantages of different approaches to communication scheduling. RTCAN takes into account time-triggered communication derived from control loops, guaranteeing temporal determinism, as well as event-triggered communication by sensors, which are transmitted with low latency. An implementation of the protocol is available as open-source software library, which can easily be ported to new platforms. Finally, results from benchmarks performed on actual hardware are reported, showing the ability of RTCAN in handling heterogeneous communications.

1 INTRODUCTION

Robots are distributed systems where different devices perform specific tasks and need to exchange data to run the overall system. The distributed approach is becoming common also to encourage hardware reuse (Bonarini et al., 2012), thus speeding up the development of new robotic systems, as it has been done since years by software developers (Bruyninckx, 2001; Quigley et al., 2009). In a distributed architecture, a key aspect is how communication requests are handled: should they be triggered by time or by events? How priorities should be assigned to different data sources? Do we need a static or a dynamic network configuration? There are many fields, e.g., factory automation, automotive networking, or sensor networks, where answers to these questions can be easily picked out, and one approach to communication scheduling can be recognized as preferable. It is not so for robotic applications, where it is hard to define which communication paradigm is the best, as different requirements are needed by the different components of a complex robotic system.

In this paper, we present a new CAN-Bus protocol focused on robotic applications, which aims at combining the best characteristics of different approaches to communication scheduling, taking in account time determinism, fast event response, and flexibility.

2 TIME-TRIGGERED AND EVENT-TRIGGERED COMMUNICATION

The choice between time-triggered and event-triggered communication paradigms to design real-time systems has been subject to a long debate (Kopetz, 1993; Obermaisser, 2004), which highlighted advantages and drawbacks of both approaches.

A time-triggered design leads to predictable temporal behavior, since each communication occurrence is planned at design time. This requires a detailed analysis of the overall system, which adds complexity to its design and limits further system expansions. On the other side, a-priori scheduling leads to temporal determinism of communication, guaranteeing quality of service. To improve flexibility, a centralized scheduler can be exploited performing online admissibility control. Depending on the scheduling policy adopted, an online scheduler can lead to high computational requirements.

On the other hand, an event-triggered design does not need a-priori knowledge about the system to schedule communication, thus leading to a much more flexible design. As a consequence of such a flexibility, a much more extensive testing is required

Migliavacca M., Bonarini A. and Matteucci M..
RTCAN - A Real-time CAN-bus Protocol for Robotic Applications.
DOI: 10.5220/0004484303530360
In *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2013), pages 353-360
ISBN: 978-989-8565-71-6

to verify that the system can handle communication requests under different load situations. An advantage of event-triggered designs is, generally, a better resource exploitation with respect to time-triggered designs, with higher achievable throughputs (Albert, 2004).

2.1 Communication Requirements in Robotic Systems

A first source of communication in a distributed robotic system are control loops, which need to exchange data between sensors and actuators. These data transfers are, generally, periodic, deterministic and known at design time. Control loops are highly affected by the presence of jitter (Pèrez et al., 2003), which introduces variable delay and, thus, may induce overshoots of the control action and instability. Another common source of data transmission is the broadcast of system status, like a heartbeat signal, to suddenly halt the system if a failure happens. The system status could be updated asynchronously on change, but updating it periodically is generally a safer solution: if the status is not received, every component of the system can enter a safe mode. This kind of periodic communication are best handled in a time-triggered way, to schedule transmission occurrences, preventing collisions and reducing jitter.

Besides the sensors used in control loops, in a robotic system, we find other kinds of data sources; for instance, proximity sensors and bumpers produce useful data only when triggered by some event, like approaching an obstacle, and the most important factor is to react to new reading as quick as possible. If sensor readings are transmitted periodically, the only way to reduce the worst case latency is to increase the update frequency, which leads to a waste of bandwidth when data are not relevant. Using event-triggered transmissions saves bandwidth and, generally, reduces latency. Other sources of non periodic data are planners, which can be triggered by some event (e.g., a new goal) and their execution may be not constant in time. Again, to save bandwidth and reduce latency, an event-triggered messaging paradigm is preferable to transmit the output of a planner.

As a conclusion, in robotic systems both periodic, time-triggered, and sporadic, event-triggered, data transmissions are needed, thus a protocol which combines the two communication paradigms is desirable.

2.2 Scheduling Time- and Event-Triggered Traffic on the CAN-bus

To connect the components of a distributed system a bus approach is preferable with respect to point-to-point architectures, reducing wiring complexity and making it easy to add devices to the network (Bonarini et al., 2011). Among the available bus communication standards, one of the most common is the CAN-Bus, which has been developed for automotive applications and is implemented by almost all microcontroller manufacturers, making it a good choice for distributed robotic systems.

The CAN-Bus features a carrier-sense multiple-access (CSMA) media access control (MAC), which is based on the ability of CAN controllers to detect the bus status while transmitting. Data are transmitted through a binary model of *dominant* and *recessive* bits; during the transmission of the arbitration field of a CAN frame, if the controller recognizes a dominant bit while it was trying to transmit a recessive one, it knows that the arbitration is lost and the node becomes a receiver. The CAN-Bus also focuses on fault detection and tolerance, providing hardware CRC calculation, automatic retransmission and many other features, as it was designed for automotive applications where reliability is a primary requirement.

Due to hardware arbitration, the CAN-Bus is well suited for fixed-priority event-triggered communication, where high-priority messages win arbitration on low-priority ones; moreover, latency depends on bus load and a high-priority transmission can always be preempted by a higher-priority one. In order to extend CAN-Bus applications to distributed control systems, where temporal determinism and low jitter are mandatory, several protocols to schedule time-triggered traffic on the CAN-Bus have been presented and reviews are available (Almeida et al., 2002; Nolte et al., 2003; Coronel et al., 2005).

As discussed in the previous section, in robotic applications we must take into account both periodic and sporadic communication, derived from distributed control loops and asynchronous events, respectively. To effectively combine the two paradigms, a key aspect is temporal isolation: event-triggered traffic should not compromise time-triggered tasks. For these reasons, in this paper we do not consider the protocols without temporal isolation, e.g., DeviceNET (Noonen et al., 1994).

Looking for a CAN-Bus protocol that satisfies the requirements of robot developers, we find two proposals that are still actively developed: TTCAN (time-triggered CAN) (Leen and Heffernan, 2002), which has been standardized by the ISO Technical Commitee (ISO 11898-4), and FTT-CAN (Flexible Time-Triggered CAN) (Pedreiras and Almeida, 2000; Almeida et al., 2002), proposed to handle time-triggered and event-triggered traffic without loosing flexibility.

TTCAN uses a time-division multiple-access (TDMA) approach: *temporal windows* are defined at design time, and each window can be reserved to a single node for time-triggered messages (exclusive windows), or left available for event-triggered messages (arbitration windows) where nodes compete for bus access using CAN-Bus CSMA arbitration. To align the local clocks on all nodes, reference messages sent by a master node are used. The interval between two reference messages is called *basic cycle*, each containing several time slots. Basic cycles are then organized in a static calendar, named *system matrix*, which repeats periodically. The system matrix has two main drawbacks that severely limit TTCAN flexibility: first of all, it is static and must be entirely known by all nodes at boot time, thus, adding a message to the system means reprogramming all the nodes; moreover, basic cycles are organized in columns, defining the length of time slots, which must be shared by all cycles, limiting the maximum throughput.

FTT-CAN is a CAN-Bus protocol which combines TDMA and CSMA techniques to handle both time triggered and event triggered traffic. FTT-CAN takes also into account dynamic scheduling, allowing online admission control and centralized scheduling on the master node, which can be useful when working with dynamic network configurations. In FTT-CAN, a master node transmits periodic trigger messages, aligning the local clocks on all nodes and defining communication rounds called *elementary cycles* (ECs). Within each EC the protocol defines two consecutive, distinct communication phases: the *asynchronous window* first, followed by the *syncronous window*. During the asynchronous window, event-triggered messages compete for the bus as soon as they request a transmission, using the CAN-Bus CSMA arbitration. The synchronous window is reserved for time-triggered messages, and its duration varies depending on the traffic scheduled within that EC. Within the synchronous window, time-triggered messages still use CSMA arbitration, meaning that there is no predefined delivery order, but the synchronous window length guarantees that all scheduled messages will be delivered during the current EC. To compute the length of the synchronous window, the master node runs a scheduler at the beginning of each EC and broadcasts the resulting duration in the payload of the trigger message. In this way, temporal isolation between event-triggered and time-triggered communication is achieved. A limit of this approach is that the minimum period for time-triggered messages is the period of the trigger message, limiting the maximum frequency of periodic traffic. Reducing the period of trigger messages leads to higher protocol overhead and lower throughput. Moreover, using CSMA arbitration within the synchronous window, the jitter of time-triggered messages is bounded only by the duration of the synchronous windows they are scheduled in, which can be long and may vary at each communication cycle. FTT-CAN is designed to allow different priority policies for CSMA arbitration (e.g., rate monotonic, deadline monotonic, earliest deadline first), but it is still not possible to achieve high temporal determinism. The windowing mechanism increases the latency of event-triggered transmissions too, as grouping together the periodic messages means that all events occurred during a synchronous windows will be delayed to the next communication cycle.

2.3 Combining Flexibility, Low jitter and Low Latency

From the analysis of TTCAN and FTT-CAN, we can conclude that TTCAN achieves better temporal determinism, transmitting time-triggered data in a pure TDMA approach, while FTT-CAN is much more flexible, with no shared and static schedule. On the other hand, TTCAN requires a rigid and static schedule which must be known by all nodes, sacrificing flexibility, while FTT-CAN is affected by jitter and latency due to the windowing system. From the point of view of robot designers, we would like to exploit the best of the two, which leads us to the proposal of a new CAN-Bus protocol: RTCAN.

3 RTCAN

RTCAN is a real-time protocol for the CAN-Bus focused on the communication requirements of a distributed robotic system: limited jitter for control loops, low latency to quickly react to events, and flexibility to easily add features, thus networked nodes, to an existing systems.

3.1 RTCAN Message Types

To support the requirements of a robotic system, which have been pointed out in Section 2.1, in RTCAN we define three distinct types of messages, each focused on the specific characteristics of their application area:

- Hard real-time messages (*HRT*) are periodic messages, e.g., from distributed control loops; they are deterministic, their deadlines are absolute in time and should never be missed.
- Soft real-time messages (*SRT*) are triggered by events, e.g. new sensor readings; they are not periodic neither deterministic, but they need to be transmitted with the lowest possible latency. Deadlines are relative and if missed the system can still operate.
- Non real-time messages (*NRT*) do not expire in time, e.g., logging messages; they can be delivered without any latency constraints, exploiting free resources when available.

HRT messages transmission is time-triggered, in a pure TDMA approach, and bus access is reserved in a calendar for each message occurrence. SRT messages are transmitted only when the bus is not reserved, not to interfere with HRT communication. Concurrent SRT messages compete using the hardware CSMA arbitration of CAN controllers, with priority assigned by Earliest Deadline First scheduling. NRT messages are handled as soft real-time ones, but their priority is fixed and always lower than SRT messages.

3.2 RTCAN Communication Cycle

As in TTCAN and FTT-CAN, RTCAN exploits periodic messages, named *sync messages*, sent by a master node to align the local clocks on all nodes. This is needed by the TDMA approach used to trigger the transmission of HRT messages. The time between two sync messages defines a communication cycle, which is in turn divided in several time slots. Each time slot can be reserved for the transmission of a HRT message (or a part of it, if fragmentation is needed) or can be available for SRT and NRT messages. Time slots are reserved by a centralized scheduler running on the master node (see Section 3.3), which sends the reservation plan for the beginning cycle to all nodes, in the payload of the sync message. The reservation plan is a simple bit mask where a 1 denotes a slot reserved to a HRT message, while a 0 means that the slot is free for the CSMA arbitration of other messages. In this way, HRT and SRT/NRT messages are handled with temporal isolation, resulting in temporal determinism. Time slots length is application dependent: it should be at least as long as an empty CAN 2.0B frame, and no longer than a full one (from 64 to 128 bit-times, plus the overhead of bit stuffing imposed by the CAN-Bus). The maximum number of time slots within each cycle is limited by the 8 bytes payload of CAN frames, which gives a temporal horizon of 64 time slots for the reservation mask. The choice of slots per cycle determines the maximum throughput: smaller slots give higher bandwidth if many small messages are sent on the bus, but as the reservation mask is limited in size the frequency of the sync messages must be increased, wasting some bandwidth.

3.3 HRT Message Scheduling

HRT messages are time-triggered, they must be always delivered in time with the lowest possible jitter. To guarantee bus access to all HRT message occurrences, time slots are reserved in a calendar by a centralized scheduler which runs on the master node. In RTCAN, HRT messages are periodic, future transmission occurrences are known a-priori, and the scheduling process reduces to admission control and phase displacement, assigning the first transmission slot to each message, to avoid temporal overlaps.

Admission control is done by checking that the requested transmission period is not relatively prime to other scheduled transmission periods. Then, an initial phase displacement, which determines the time slot for the first transmission, is assigned by the scheduler to the HRT message. Messages sent at higher rate, i.e., with lower period, limit the number of available phase displacements, thus reducing the number of schedulable messages. This simple approach to HRT messages scheduling restricts the range of concurrent scheduled periods, but removes transmission jitter (besides variable latency of the communication media, which is negligible on the CAN-Bus). Moreover, having control loops which run at periods which are relatively multiple is common on robotic systems, then such a limitation is, generally, not a problem.

Given the initial phase displacement, each node can compute the next reserved time slot for a HRT message simply adding the period. In other words, the scheduler and the admission control are centralized on the master node, but the reservation calendars are local. Only the master node needs to know all HRT message reservations, updating the global calendar to generate the reservation mask sent with the sync message; this is required to avoid event-triggered transmission attempts in reserved time slots. Using a centralized scheduler also facilitates online dynamic scheduling, preserving flexibility.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

000000	HRT Message ID	Fragment #
Laxity	SRT Message ID	Fragment #
111111	NRT Message ID	Fragment #

Figure 1: RTCAN priority and message ID encoded into the CAN bus 29-bit Extended ID.

3.4 SRT and NRT Message Arbitration

SRT messages are triggered by events, their delivery latency should be low but they must not interfere with HRT communication. They are sent only in slots marked as available in the reservation mask, and compete for the bus using the CAN hardware arbitration. In order to reduce latency, while enhancing resource exploitation, deadline-based dynamic schedulers are preferable with respect to fixed priority schedulers or Rate Monotonic schedulers (Buttazzo, 2005; Lu et al., 2002). The scheduling policy we have adopted for soft real-time messages is inspired by Earliest Deadline First (*EDF*) and Least Laxity First (*LLF*) schedulers: a SRT message increases its priority while it is approaching its deadline (Livani and Kaiser, 1998; Kaiser and Livani, 1998). To exploit the CAN-Bus carrier-sense multiple-arbitration, the laxity of the message is encoded in the first bits of the CAN frame arbitration field, as shown in Figure 1. In this way messages near their deadline have higher chances to be transmitted with respect to messages with far deadlines thanks the hardware arbitration. Laxity can be encoded linearly or using a logarithmic scale, resulting in a finer resolution for nearer deadlines, and a coarser resolution for further deadlines (Natale, 2000). The remaining bits of the arbitration field include an ID to identify RTCAN messages, and a fragment counter used to handle messages longer than the payload of a CAN frame, as explained in Section 3.5.

NRT messages are handled as SRT ones, but their transmission priority is always lower (the laxity bits are all recessive) and it is never increased, thus they always loose the arbitration against SRT messages.

3.5 Fragmentation

Fragmentation is handled by RTCAN as well; payloads longer than 8 bytes, which is the maximum payload of CAN frames, are fragmented and transmitted in sequence. A fragment counter in the CAN arbitration field identifies fragmented messages, and they are concatenated during the reception. This simple approach exploits the CAN-Bus guarantee that packets are received in the same order as they are transmitted. All RTCAN messages can be fragmented: HRT ones will span over more reserved time slots, while SRT ones must deliver the last fragment before their deadline.

RTCAN is focused on real-time communication and not on fault tolerance: receive errors (e.g., overruns or checksum mismatches) are not handled by RTCAN: the message is just signed as corrupt and retransmission requests should be implemented by a higher layer protocol.

3.6 Open Source Implementation

RTCAN has been implemented as an open source software library, with particular attention to portability, in order to facilitate its extension to new platforms. The library is organized in two layers: the high level sources provide the API and implement the HRT scheduler, the fragmentation mechanism and the platform-independent code of interrupt handlers; the low level drivers interface with the peripherals and abstract interrupt requests. To port RTCAN to new platforms, only the low level layer needs to be implemented, writing the drivers for the specific CAN controller and for the hardware timer used to handle time-triggered operations.

RTCAN open source library is available on `http://github.com/openrobots-dev/RTCAN`; current implementation supports STM32 ARM Cortex-Mx microcontrollers and the ChibiOS/RT real-time operating system (Di-Sirio, 2007).

4 BENCHMARKS

RTCAN have been tested on the hardware nodes shown in Figure 2 and some benchmarks have been produced to evaluate its communication performance. In the following sections, the experimental setup is described and benchmark results are reported.

4.1 Experimental Setup

RTCAN tests have been conducted on custom modules featuring STM32F103RET6 microcontrollers (ARM Cortex-M3 at 72Mhz, 20kb of RAM). To evaluate the protocol, four nodes have been used: three of them communicate using RTCAN, while one more module captures and timestamps with microsecond precision all transmitted packets. During the tests the cycle period is set to 10 ms and the number of slots per cycle is 60. The CAN controllers are configured

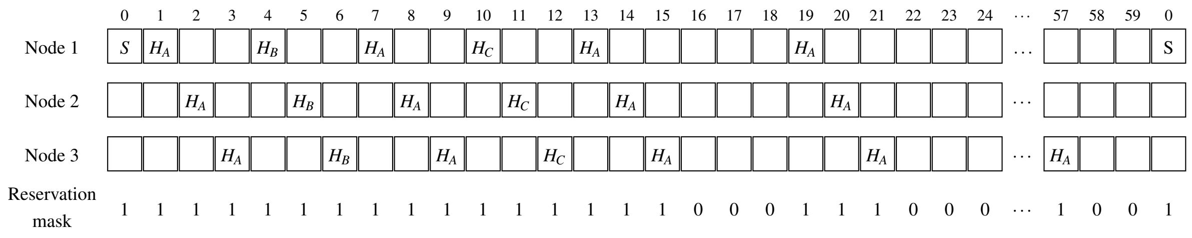


Figure 3: The HRT calendar returned by the scheduler with messages used in the benchmarks and the respective reservation mask. H_A messages have 1 ms transmission period, H_B 5 ms and H_C 10 ms. S is the sync message from the master node.

Figure 2: The custom hardware modules used in the benchmarks.

for 1 Mbit data rate. As a consequence, each time slot lasts 166 µs, which can contain a 8 byte CAN frame with worst case bit stuffing (158 bit times), plus the CAN intermission field (3 bit times) and 5 µs of slots separation. A fixed slot length of 166 µs introduces a discretization of admissible frequencies, thus the actual frequency obtained could be slightly different from the required one (e.g., 1000 Hz is translated to a period of 6 time slots, which leads to an actual frequency of 996 Hz). Anyway, differences are, generally, small, and we are much more interested in guaranteeing temporal determinism and low jitter than precise transmission frequencies.

Table 1: HRT time-triggered messages scheduled on each node.

ID	#	Period [ms]	Payload [bytes]	Throughput [kbps]
H_A	1	1	8	64
H_B	1	5	8	12.8
H_C	1	10	8	6.4

Table 2: SRT event-triggered message activity on each node.

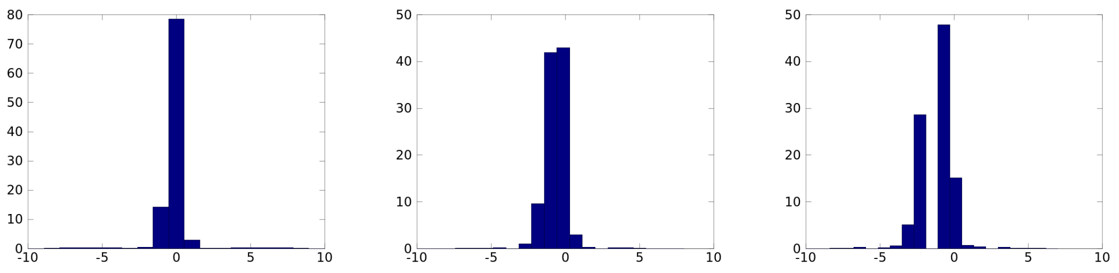
ID	#	Deadline [ms]	Event Period [ms]	Payload [bytes]
S_A	5	10	10–20	8
S_B	5	10	20–50	16
S_C	5	50	50–100	20

With the test configuration, the maximum throughput is 378 kbps, which means only a little overhead with respect to the maximum data rate of CAN 2.0B specifications, which is 398 kbps (again, in worst-case bit stuffing situation). Having 60 time slots per cycle, the maximum frequency for HRT messages is 3 KHz, due to the fact that slot 0 is reserved to the sync message. To benchmark RTCAN, each of the nodes schedules 3 HRT messages as reported in Table 1; this gives a total time-triggered traffic of 249.6 kbps leaving some resources to SRT and NRT messages. Given the HRT requirements of the nodes, the centralized scheduler produces a reservation mask to reserve bus access to all time-triggered messages occurrences. The resulting communication cycle, which in this configuration is periodic, due to the requested frequencies, is presented in Figure 3.

To exploit the remaining bandwidth, SRT messages are produced on each node, simulating event-triggered traffic. The trigger period varies, while the relative deadline is fixed for each message. The test configuration for SRT messages is shown in Table 2; the resulting average traffic is 150.9 kbps. As a consequence, the total requested throughput is about 400 kbps, higher than the maximum admissible value of 378 kbps.

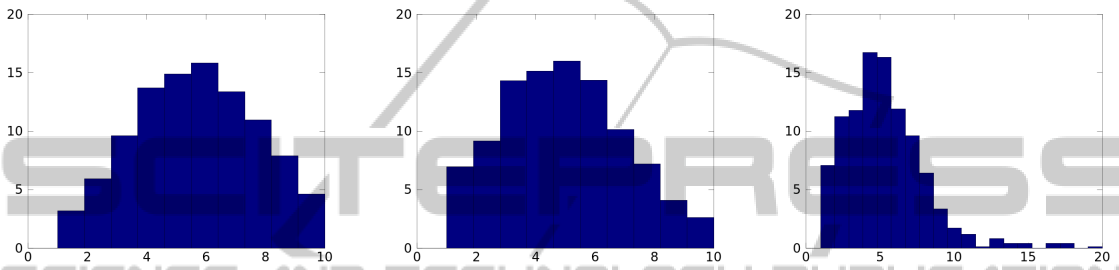
4.2 Results

During the benchmarks about 250.000 HRT and 100.000 SRT messages were transmitted. With respect to HRT messages, we evaluated the actual update period, its standard deviation and the distribution of transmission jitter. Results are reported in Table 3, while Figure 4 shows the distribution of transmission jitter. The results show that HRT messages never missed a deadline and the jitter is bounded to ±3 µs. Moreover, due to temporal separation, HRT performances are not influenced by the actual bus load. About the measured periods, we believe that some unavoidable jitter is introduced by the CAN itself, which adapts the bit time to other nodes, in order



(a) 1 ms period (b) 5 ms period (c) 10 ms period

Figure 4: Distribution of HRT messages transmission jitter, expressed in µs.



(a) 10–20 ms period, 10 ms deadline (b) 20–50 ms period, 10 ms deadline (c) 50–100 ms period, 20 ms deadline

Figure 5: Distribution of SRT messages transmission latency expressed in ms.

to have consistent bit timing over the bus. This feature cannot be deactivated on our CAN controllers and it is not measurable without specific network analyzers.

For SRT messages, we evaluated the missed deadline ratio and the delivery latency. Missed deadline ratio and mean latency are reported in Table 4, while latency distribution is shown in Figure 5. The results show that S_C messages, which have longer deadlines, thus a lower initial transmission priority, rarely win arbitration when the laxity is still high due to the EDF scheduling, and higher priority messages, i.e., with shorter deadlines, have less delivery latency. It can be noticed also how S_B messages, which have the same deadline of S_A messages, have a higher miss ratio, due to their fragmentation: two frames must win arbitration for a successful delivery.

The test shows that RTCAN can handle high frequency HRT messages with low jitter and a very little overhead, given by the sync message which occupies $1/60$ of the bandwidth. As a comparison, consider that to transmit a 1 Khz message in FTT-CAN, a 1 Khz trigger message is needed, wasting $1/6$ of the available bandwidth. The jitter is lower too, as in FTT-CAN it is only bounded by the length of the synchronous window. Compared to TTCAN, the centralized scheduler used in RTCAN, with only local calendars on the nodes, gives the same temporal determinism of pure TDMA protocols, but without affecting flexibility as using a static system matrix.

Table 3: HRT messages mean period and its standard deviation.

ID	Message count	Measured period [µs]	Standard devation [µs]
H_A	61136	995.9	0.86
H_B	12427	4979.4	0.93
H_C	6213	9958.8	1.05

Table 4: SRT messages missed deadline ratio and mean transmission latency.

ID	Message count	Missed ratio	Mean latency [ms]
S_A	4121	1.60%	4.33
S_B	1594	5.52%	5.13
S_C	784	5.21%	14.94

5 CONCLUSIONS

In this paper we discussed the communication requirements of robotic systems, highlighting that both time-triggered and event-triggered communications are needed. To account both communication paradigms we developed RTCAN, a real-time CAN-Bus protocol designed for distributed robotic applications; RTCAN combines the best aspects of CSMA and TDMA protocols without scarifying flexibility. Benchmarks show that RTCAN is able to handle HRT messages at high frequency, with high temporal determinism and low jitter, by using temporal separation. At the same time, SRT messages are delivered exploiting free resources with a scheduling policy which guarantees high bandwidth exploitation and limits latency. Moreover, a ready to use implementation is available as open-source software library, and its layered architecture facilitates porting to new platforms.

RTCAN has been used to control *Triskar2*, an omnidirectional wheeled robot built from distributed hardware modules. Three motor controllers drive the wheels, while another board reads measurement from IR proximity sensors. The distributed motion control loop exploits HRT messages to synchronize motor actuation with low jitter; the proximity module sends SRT messages when an obstacle is detected. Thanks to RTCAN flexibility, different data sources, with different requirements, are mixed on the CAN-Bus.

ACKNOWLEDGEMENTS

This work has been partially supported by the research grant "Robotics for the Masses" from ST Microelectronics and Regione Lombardia, and by the Italian Ministry of University and Research (MIUR) through the PRIN 2009 grant "ROAMFREE: Robust Odometry Applying Multi-sensor Fusion to Reduce Estimation Errors".

REFERENCES

Albert, A. (2004). Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. *Embedded World*, 2004:235–252.

Almeida, L., Pedreiras, P., and Fonseca, J. A. G. (2002). The FTT-CAN protocol: why and how. In *IEEE Transactions on Industrial Electronics*, pages 1189–1201. IEEE Press.

Bonarini, A., Matteucci, M., Migliavacca, M., and Rizzi, D. (2012). R2P: an open source modular architecture for rapid prototyping of robotics applications. In *Proceedings of 1st IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control (CESCIT'12)*, pages 68–73. Elsevier.

Bonarini, A., Matteucci, M., Migliavacca, M., Sannino, R., and Caltabiano, D. (2011). Modular low-cost robotics: What communication infrastructure? In *Proceedings of 18th World Congress of the International Federation of Automatic Control (IFAC)*, pages 917–922. Elsevier.

Bruyninckx, H. (2001). Open robot control software: the OROCOS project. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, pages 2523–2528. IEEE Press.

Buttazzo, G. C. (2005). Rate monotonic vs. EDF: judgment day. *Real-Time Systems*, 29:5–26.

Coronel, J., Blanes, F., Benet, G., Simó, J., Pèrez, P., and Albero, M. (2005). CAN-based distributed control architecture using the SCoCAN communication protocol. In *Prodeedings of 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE Press.

Di-Sirio, G. (2007). ChibiOS/RT Real Time Operating System. `http://www.chibios.org`.

Kaiser, J. and Livani, M. A. (1998). Invocation of real-time objects in a can bus-system. In *Proceedings of the The 1st IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 298–307. IEEE Press.

Kopetz, H. (1993). Should responsive systems be event-triggered or time-triggered? *Transactions on Information and Systems*, E76-D(11):1325–1332.

Leen, G. and Heffernan, D. (2002). TTCAN: a new time-triggered controller area network. *Microprocessors and Microsystems*, 26(2):77–94.

Livani, M. A. and Kaiser, J. (1998). EDF Consensus on CAN Bus Access for Dynamic Real-Time Applications. In *Proceedings of 12th International Parallel Processing Symposium and 9th Symposium on Parallel and Distributed Processing*, volume 1388, pages 1088–1097. Springer.

Lu, C., Stankovic, J., Son, S., and Tao, G. (2002). Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real-Time Systems*, 23:85–126.

Natale, M. D. (2000). Scheduling the can bus with earliest deadline techniques. In *Proceedings of The 21st IEEE Real-Time Systems Symposium.*, pages 259–268. IEEE Press.

Nolte, T., Nolin, M., and Hansson, H. (2003). Server-based scheduling of the can bus. In *Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'03)*, pages 169–176. IEEE Press.

Noonen, D., Siegel, S., and Maloney, P. (1994). DeviceNet Application Protocol. In *1st International CAN Conference*.

Obermaisser, R. (2004). *Event-triggered and time-triggered control paradigms*, volume 22 of *Real-Time Systems*. Springer.

Pedreiras, P. and Almeida, L. (2000). Combining event-triggered and time-triggered traffic in FTT-CAN: Analysis of the asynchronous messaging system. In *IEEE International Workshop on Factory Communication Systems*, pages 67–75.

Pèrez, P., Benet, G., Blanes, F., and Simó, J. (2003). Communication jitter influence on control loops using protocols for distributed real-time systems on CAN bus. In *Proceedings of 5th IFAC International symposium SICICA*, pages 237–243. Springer.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*. IEEE Press.