A Real-time Wireless Communication System based on 802.11 MAC

Gennaro Boggia, Pietro Camarda, L. Alfredo Grieco and Giammarco Zacheo
DEE – Politecnico di Bari
Italy

1. Introduction

IEEE 802.11 Wireless Local Area Networks (WLANs) are very pervasively deployed in the consumer market, due to their ability to provide ubiquitous network access with high flexibility, cheap costs, and ease of installation and maintenance. In such networks, the wireless channel is shared among the stations and, in order to deal with packet collisions, a Carrier Sense Multiple Access with Collision Avoidance algorithm is employed. The scientific community is now investigating in which measure this technology can be exploited also in real-time contexts, where several tasks have to be served with a high degree of determinism to provide the expected service, such as in Networked Control Systems (NCSs). This issue is challenging because the network interface cards available on the market have been mainly conceived to provide a best effort service, without any guarantees on packet delay. Moreover, the unpredictable behavior of the radio channel further worsen the problem.
NCSs exploit a packet switching network to connect spatially distributed sensors, actuators, and controllers (Hespana et al., 2007). In this way, a NCS can accomplish complex control tasks without requiring cumbersome wiring infrastructures. In fact, point-to-point interconnections are replaced by a communication network, which is shared among all the components of the control system using statistical multiplexing. A key issue of NCSs is that the Quality of Control of the system is influenced by the Quality of Service of the underlying communication system (Buttazzo et al., 2007; Baillieul & Antsakls, 2007). To solve the problem three different class of techniques can be jointly adopted:

1. an advanced control design to counteract time-varying packet delays and losses induced by network (Schenato et al., 2007; Nair et al., 2007; Liu et al., 2007; Wu & Chen, 2007);
2. the use of Real Time Operating Systems (RTOSs) at NCS nodes to timely serve events scheduled by control applications (Baillieul & Antsakls, 2007, Kim et al., 2006);
3. the reduction as much as possible of network delays and packet losses (Hespana et al., 2007; Boggia et al., 2008a; Boggia et al., 2008b).

Until now, these topics have been mainly investigated with reference to wired NCSs. But, the trend is toward wireless communications because they can further reduce wiring and they can increase the flexibility of a NCS, giving the chance to build, on the fly, control systems made be sensors, actuators and controllers placed in the same area (Baillieul & Antsaklis, 2007; Moyne & Tilbury, 2007; Burda & Wietfeld, 2007). Anyway, it is not straightforward to build a wireless networked control system (WNCS) due to the unpredictable behavior of the radio channel (Walke et al., 2006; Cena et al., 2007). Starting from this premise, the present chapter focuses on three main objectives:

i. to analyze the state of the art on wireless real-time systems;
ii. to experimentally evaluate the performance bounds of a 802.11-based wireless real-time communication platform;
iii. to provide guidelines to design an effective TDMA (Time Division Multiple Access) strategy for wireless real-time systems starting from the so derived performance bounds and integrating the leading IEEE 802.11 technology (Walke et al., 2006), the RTnet framework (Kiszka et al., 2005a), and the Xenomay nanokernel (Gerum, 2004).

The rest of the chapter is organized as follows: Sec. 2 summarizes related works on real-time wireless communications technologies. Sec. 3 provides an overview on 802.11 WLANs and RTOSs, as basic elements to realize a wireless real-time system. In Sec. 4, the architecture of a WNCS is described and its performances are evaluated in Sec. 5. In Sec. 6, the design guidelines for a TDMA scheme based on the considered architecture are given. In Sec. 7, the performance of the proposed TDMA scheme are experimentally evaluated. Finally, the last section outlines the conclusions.

2. State of the art on wireless real-time systems

In recent years, research activities in the field of wireless real-time technologies for NCSs have been very active as testified by the relevant amount of literature produced on this subject, by the number of developed simulation/experimentation platforms (Andersson et al., 2005; Cervin et al. 2007; Hasan et al., 2007; Nethi et al., 2007; Biasi et al., 2008; Chen et al., 2008), and by the already available communication technologies (Neumann, 2007; Pellegrini et al., 2006; Willig et al., 2005) for WNCSs.

Herein, it follows a review of the most important contributions that are related to our discussion. In particular, we will focus on the most recent ones, leaving the reader to consult the excellent surveys (Neumann, 2007; Pellegrini et al., 2006; Willig et al., 2005) to gain a full vision of the world of wireless real-time networks.

In (Flammini et al., 2009), a novel wireless real-time communication protocol has been designed and experimentally evaluated. It exploits standard hardware and implements a hybrid medium access strategy. Time Division Multiple Access scheduling is used to ensure time deadlines respect, while Carrier Sense Multiple Access with Collision Avoidance is used for acyclic communications. It has been successfully tested in a prototype network that adopts star topology and can manage up to 16 nodes with a refresh time of 128 ms.

In (Heynicke et al., 2008; Krber et al., 2007), a gateway to interconnect hybrid wireless/wired control networks is proposed. The gateway is based on standard equipments such as the Chipcom CC2400 device by Texas Instruments. Its effectiveness has been demonstrated in an experimental testbed made by 32 nodes handled using four frequency channels and eight time-slots per channel.

In (Boughanmi et al., 2008), the suitability of IEEE 802.15.4 Wireless Personal Area Networks (WPANs) (IEEE, 2006) for wireless networked control systems has been investigated. In particular, using the TrueTime Matlab/Simulink simulator (Cervin et al., 2007), it has been shown that the joint adoption of the beacon-enabled mode and of the Guaranteed Time Slot mechanism can allow the support up to two control loops with sampling periods not smaller than 15.36 ms. Analogously, in a less recent work (Choi et al., 2006), a wireless real-time network based on the 802.15.4 MAC has been designed, which is able to satisfy deadlines not smaller than 100 ms.

In Sep. 2007, the WirelessHART standard has been issued (Song et al., 2008) with the objective to support process measurement and control applications. WirelessHART is a secure, low-speed, if compared to 802.11g WLANs (IEEE, 1999a), and TDMA-based wireless mesh networking technology. It uses a central network manager to pro-vide routing and communication schedules. At the very bottom, it adopts the IEEE 802.15.4 physical layer and operates in the 2.4 GHz ISM radio band using 15 different channels (Biasi et al., 2008). WirelessHART appears a promising technology in this field and research activities are on going to assess its performance bounds (Biasi et al., 2008).

In (Lee et al., 2008), in order to improve the real-time performance and reduce the transmission delay of IEEE 802.11b WLANs, a four-layer architecture has been proposed and experimentally tested, based on the network driver interface specification (NDIS) (Floroiu et al., 2001) combined with a virtual scheduling algorithm that avoids collisions. In a network scenario with nine nodes, it has been shown that the architecture is able to provide an upper bound on packet delay comprised between 10 ms and 20 ms, depending on the network conditions.

In (Robinson & Kumar, 2007), the problem of selecting what information should be sent between a sensor and a controller in a networked control system where the two components are separated by an unreliable, bandwidth limited communication link, such as a wireless one, has been analyzed. It has been shown that the common practice of sending the most recent observation is not optimal. Moreover, necessary and sufficient conditions for the existence of a combination of past and present measurements that minimizes the state error covariance have been derived. These results could have serious implications in the design of future generation highlevel protocols that modify the contents of packets waiting to be sent by taking into account the status of the previous transmissions.

In (Baliga & Kumar, 2005), the focus has been moved on the issue of middleware for networked control systems which feature the convergence of control with communication and computation. In particular, it has been shown that a software architecture able to integrate the heterogeneous technologies that compose a complex NCS is required. Moreover, the Etherware middleware is proposed and experimentally tested using a vehicular control testbed.

In (Rauchhaupt, 2002), the R-FIELDBUS project, supported by the European Commission in the 5th FP, is described. It is aimed at the implementation of a wireless fieldbus based on the architecture of Profibus DP (Pellegrini et al., 2006). An important result of the R-FIELDBUS project is the accurate investigation carried out on the available radio technologies. As a result, the IEEE 802.11b physical layer, using direct sequence spread spectrum (DSSS) modulation, was selected as the most suitable for industrial applications. Moreover, the adoption of the IEEE 802.11 MAC layer was not recommended because of the randomness possibly introduced in the packet delay. For such a reason, the R-FIELDBUS makes use of the Profibus data link layer.

3. Basic elements for a 802.11-based wireless real-time platform

3.1 The 802.11 wireless LANs

The widespread deployment of IEEE 802.11 WLANs is mainly due to their easy installation, flexibility and robustness against failures (IEEE, 1999a; Varshney, 2003). They allow the transmission at a data rate up to 54 Mbps. The 802.11 Medium Access Control (MAC) employs a mandatory contention-based channel access scheme called Distributed Coordination Function (DCF), based on the Carrier Sense Multiple Access with Collision Avoidance mechanism (Walke et al., 2006).

The fundamental building block of 802.11 architecture is the Basic Service Set (BSS), composed by a group of stations, in the same geographical area, that access the radio channel under the control of DCF. The standard defines different topological configurations, but here we will consider only the Independent BSS, which allows direct communications among stations in the same area (i.e., an ad-hoc network architecture).

Using DCF, for each frame, a station listens to the channel before transmitting. If the channel is sensed idle for a minimum interval time called DCF Interframe Space (DIFS), then the station transmits immediately the frame. Otherwise, if the medium is sensed busy, the transmission attempt is deferred until the expiration of a backoff time. Such a time is a multiple of a *slot time* (depending on the physical layer implementation), where the multiple is a random integer uniformly distributed in the interval [0, CW] and CW is the so called *Contention Window*. The *CW* size is subject to minimum and maximum bounds, CW_{min} and CW_{max}, respectively (Walke et al., 2006).

Each correctly received frame is acknowledged by an ACK frame, that is sent after a Short Interframe Space period (shorter than DIFS), to avoid that other stations use the channel. If the transmission is not successful (i.e., no ACK frame is received by the sending station), the listen-before-talking protocol and the backoff procedure are repeated by doubling the CW value up to the maximum limit of 1023 time slots.

The Collision Avoidance is obtained by the Virtual Carrier Sensing: in the header of each delivered frame there is a duration field, which indicates the time required by a transmission. The duration field is used by each station in the BSS to update the Network Allocation Vector, that accounts for the duration of the current transmission after which the channel can be sensed again for the access (Walke et al., 2006).

When a station senses a busy channel during the backoff period, its timer is frozen and the countdown is resumed after the channel is sensed idle again for a DIFS interval. This mechanism allows stations with larger backoff periods to access the channel in the next contention period. Consequently, the bandwidth allocation is more fair (Walke et al., 2006).

3.2 Background on Real-time Operating Systems

Real time operating systems (RTOSs) play a major role in NCSs. In fact, they timely serve events scheduled by control applications. Herein, we describe some basic principles about RTOSs that are relevant for our framework. A RTOS is a multitasking Operating System (OS) able to guarantee its services with deterministic response times. Several commercial (Barr, 2003) and open source (Massa, 2002) RTOSs are available, some of which are widely used and supported (Massa, 2002). In particular, in the open source community, there was a lot of effort to give hard real-time capabilities to the Linux kernel. Various projects are very active in this field and widely used in academic and industrial environments: among other solutions, there are the RT PREEMPT kernel patch (McKenney, 2005), which modifies the preemption mechanisms and interrupt handlers inside the kernel to achieve soft real-time requirements, and the RTAI (DIAPM, 2008) and Xenomai (Gerum, 2004) projects, which add a high priority scheduler for real-time tasks running concurrently with the Linux kernel with minimal intervention on the kernel itself. While the former approach allows seamless integration of the real-time tasks using standard APIs (Application Programming Interfaces), it can generally guarantee only soft real-time requirements, given that the real-time task scheduling can be influenced by system events. On the other hand, co-kernel scheduling performance is less influenced by external events, thus satisfying sharper real-time requirements.

3.3 Xenomai

Xenomai is based on the Adeos framework, which is able to provide a flexible environment for sharing hardware resources among multiple operating systems, or among multiple instances of a single OS (Yaghmour, 2001). In order to make multiple kernels run safely in parallel on the same platform, Adeos provides an efficient share to critical hardware resources, giving the opportunity to set priorities to each domain, i.e., each OS running on the machine. The original framework was intended to be a layer for machine virtualization and distributed computing environments. With its simplified version, known as I-Pipe (Interrupt Pipeline), the design has been changed to meet deterministic latencies in interrupt handling, which is fundamental for real-time operations.

Xenomai includes a core components which provides base funcions for realtime operations, which acts as a nanokernel, that is, it relies on the Linux kernel for everything it is not able to do by itself. The main components and functionalities of Xenomai are decribed below.

- **Nucleus**: it is the core of the RTOS, being responsible for thread creation, scheduling, synchronization, memory allocation, and interrupt management (Silberschatz et al., 2004).
- **RTDM**: The Real-Time Driver Model (RTDM) is a framework for real-time driver design, including file and socket I/O and descriptor handling (Kiszka, 2005b).
- **High resolution timers**: Another important feature of Xenomai is the support for high resolution timers, which provide microsecond precision for event scheduling.
- **Skins**: Emulation layers for existing RTOs (VxWorks, pSOS+, VRTX, RTAI, uITRON) (Barr, 2003).

The Native skin of Xenomai is an API with the following main characteristics:

- **Real-time IPC**: a complete set of Inter-Process Communication (IPC) primitives is available; among others, it includes message queues, one-to-one message passing and pipes, the latter providing a latency-free communication channel between real-time tasks and standard non real-time processes (ANSI, 1994).
- **Synchronization**: most of the classical synchronization objects are implemented within the Native skin, such as Mutexes, Counting Semaphores, Event flags and Condition Variables (Silberschatz et al., 2004).
- **Memory**: when dynamic memory allocation is needed by real-time tasks, Xenomai uses a region of memory whose size is selected at startup. The allocation occurs in a time-bounded fashion, and it can be used to map shared memory objects between kernel and user space.

3.4 RTnet, a hard real-time networking framework

RTnet is a hard real-time network protocol stack built on top of Xenomai and RTAI (real-time Linux extensions) (Kiszka, 2005a). It uses standard Ethernet hardware; in particular, it supports several popular network interface card chip sets, including Gigabit Ethernet and an experimental support for IEEE 802.11 WLAN. Currently, only wireless Ralink RT2500 chipset (Ralink, 2006) is supported in RTnet.

RTnet provides a POSIX socket API to real-time user space processes and kernel modules (via the RTDM interface). The main components of RTnet are the stack manager, which deals with the management of incoming packets, and the drivers, which communicate with the hardware. The stack manager is a real-time task with the highest priority: it demultiplexes the incoming packets passing their references to the respective handlers, according to the protocol which the packet belongs to.

The stack manager and the device drivers use a structure named *rtskb* to exchange and store temporarily the packets. Its use is similar to the *sk_buff* structure used in the Linux networking stack (Benvenuti, 2006). The buffer size is statically set to the network MTU (Maximum Transmission Unit), and a fixed number of buffers is allocated and assigned to each component at startup. Each component receiving a *rtskb* filled with a packet gives back an empty one, otherwise the packet is held back. Thus, at the end of each transaction each component holds the same number of *rtskb*.

Other modifications have been done in the UDP/IP implementation, in the routing, and in the IP fragmentation, which are not covered here not being related to our discussion.

RTnet has also an intermediate software layer between UDP/IP and drivers, called RTmac. Such a layer is not mandatory, but it is used for some useful services that RTnet can offer, e.g., automatic configuration (via the RTcfg module) or non real-time traffic tunneling by using virtual interfaces. Moreover, a medium access mechanism inside the RTmac can be defined, to guarantee collision-free and time-bounded transmissions. Those mechanisms are called disciplines, and can be dynamically loaded and unloaded. RTnet comes with a flexible TDMA scheme with a static master/slave approach. It provides clock synchronization, multiple slot assignments, and static transmission priorities. It is designed for wired operation and includes a fine-grained calibration mechanism which mitigates the negative effects of the transmission jitter.

The TDMA mechanism used by RTnet is based on a master-slave approach (Kiszka, 2005a). The master station manages the synchronization, and other stations can act as backup masters in order to compensate a possible master failure. The master station sends regularly a *synchronization* packet, which marks the starting of a transmission TDMA frame. Every frame is divided into transmission timeslots, and it has a numerical identifier which is incremented when a new frame starts. Each station may have one or more slots assigned for transmission, with a separate packet queue for each timeslot. It is also possible to assign multiple slots to each output queue. Each timeslot can be shared between multiple station, which can use it in turn. In Fig. 1 there is a sample configuration for a TDMA cycle. The master station sets the cycle time, while slave stations must configure their transmission slots accordingly. The configuration may be present on each station as a text file or may be distributed by the master station to all slave participants before starting the calibration phase using the RTnet configuration service, called RTcfg. A timeslot assigned to a transmitting station should be specified as an offset relative to the start of the TDMA frame (that is, with respect to the transmission of the synchronisation packet by the master).

When a slave station joins the network, it begins a synchronisation procedure with the master station, which is repeated several times in order to estimate the average transmission delay. The number of repetitions depends on the expected variance of the measurements and has to be chosen appropriately.

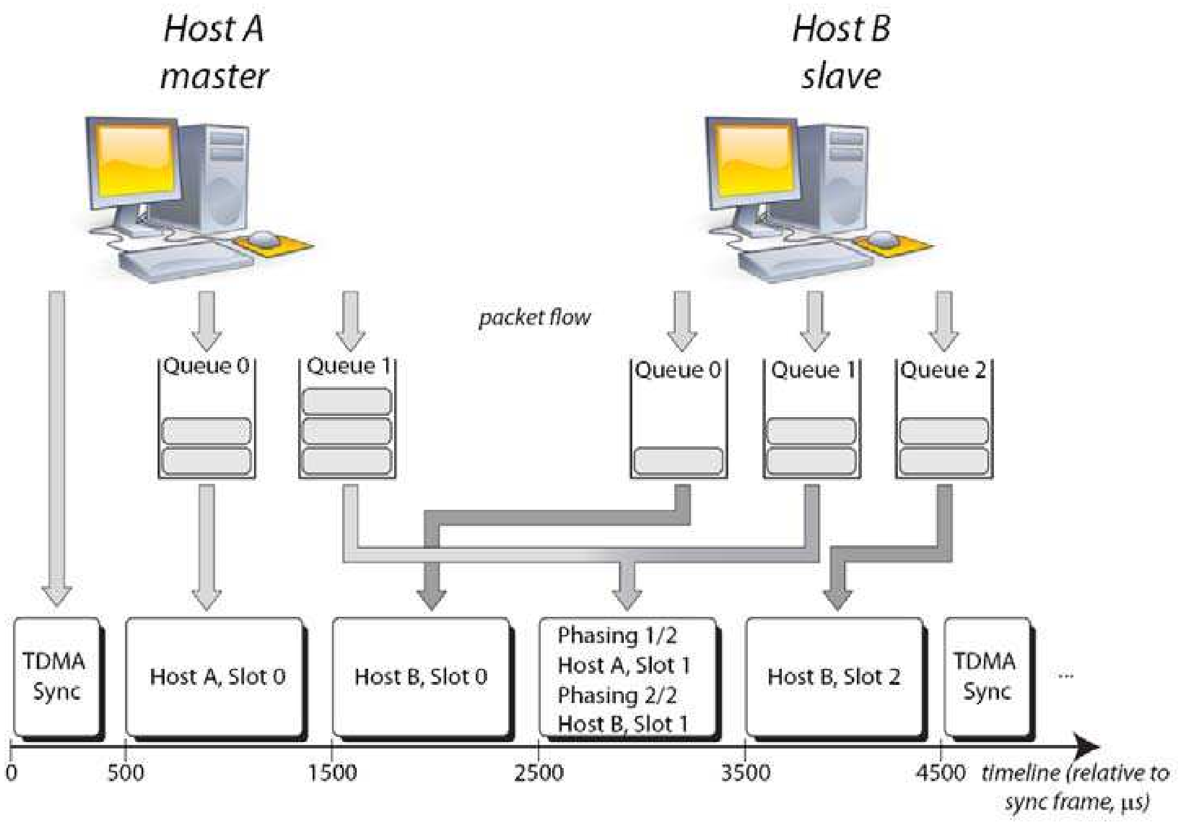


Fig. 1. Example of configuration for the structure of the TDMA frame in RTnet.

When working with wireless devices, most of the 802.11 mechanisms for collision avoidance are not used in RTnet in order to achieve a more deterministic behavior. The bit rate is statically assigned and no rate adaptive mechanisms are provided in the real-time driver. The Minimum and Maximum values for the 802.11 Contention Window can be altered on a per-packet basis or globally; also, the values for interframe spacings and the *slot time* can be altered with respect to the ones provided by the standard (IEEE, 1999a; IEEE, 1999b; IEEE, 1999c; IEEE, 2003; IEEE, 2005). The 802.11 acknowledgment mechanism can be used for signaling a successful frame delivery (ack mode), or can be completely disabled (raw mode); if the acknowledgement frame is not received the adapter considers the sent frame lost and may decide to retransmit it. The maximum of retries can be chosen by the user. It is possible to filter incoming broadcast and multicast packets, in order to reduce processing overhead for unwanted information. The last parameter that can be chosen is the baseband processor sensitivity, S_{BP}, from which the power threshold level for the receiver is computed. This threshold value is used to detect activity on the radio channel and is computed by adding an offset to the S_{BP} value. Thus, a low value for the sensitivity parameter means a low threshold, so that the receiver will be more selective during the channel sensing (this increases the probability of waiting for the channel to become idle). On the other hand, if the sensitivity value is too high, the receiver would signal that the channel is idle even if there are interferences or too much noise (thus increasing the probability of transmission errors).

4. The Architecture for a Wireless Networked Control System

Considering the previous section, we are able now to describe a system architecture that should be considered a reference framework for realizing a WNCS. Such an architecture is a real-time communications system based on Xenomai and RTnet (see Fig. 2): an application in user space may have one or more execution threads, which in turn can be real-time or non real-time. For real-time operations, a task must use the system calls of Xenomai. A real-time task wishing to use Linux system calls or non real-time library functions loses its priority and it is scheduled as a normal Linux thread until the end of the execution of the non real-time service requested.

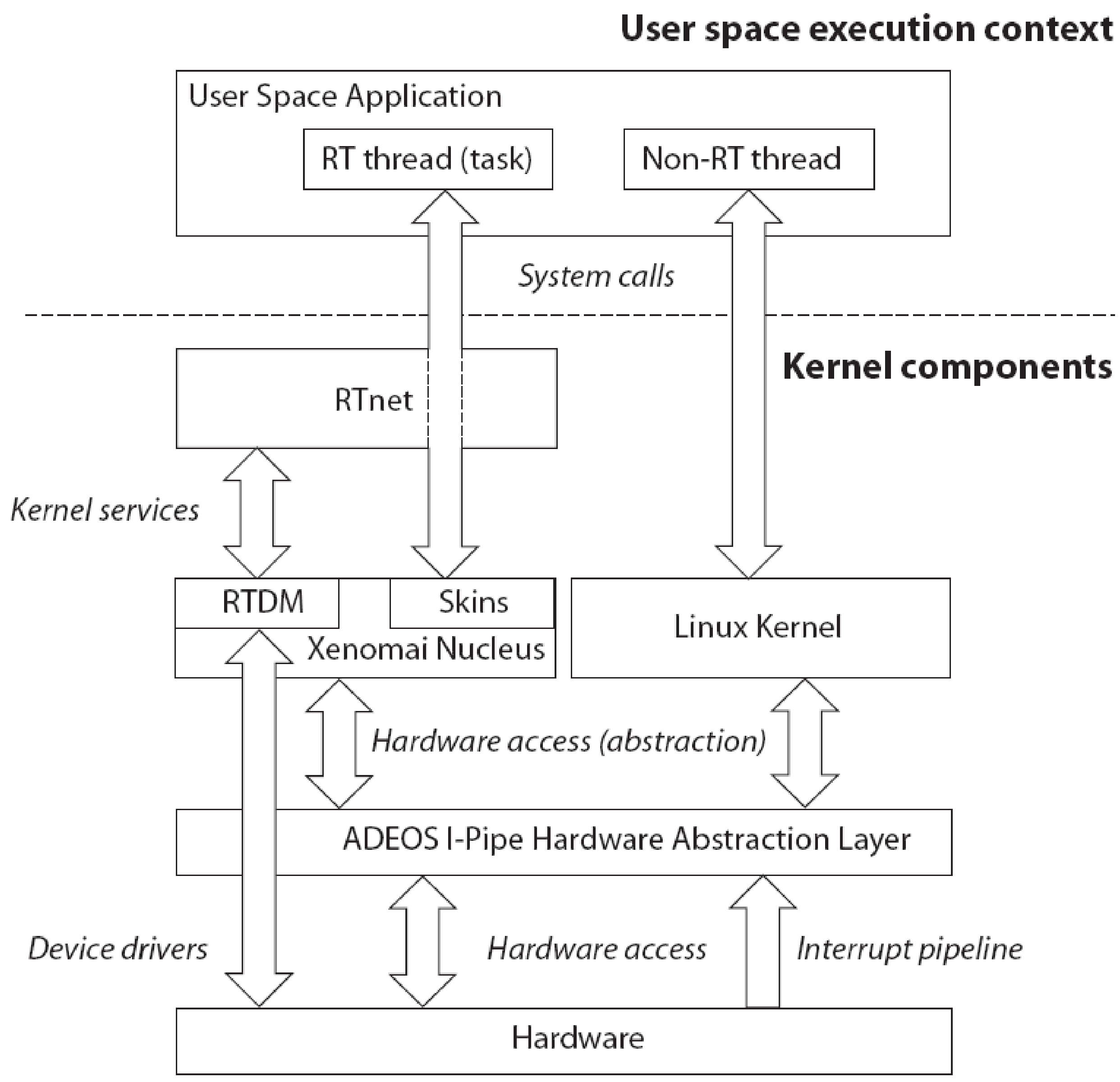


Fig. 2. The RTnet framework and its interaction with other software components.

RTnet runs in Xenomai kernel space, i.e., it is part of the kernel itself, providing network facilities to the user applications by means of a set of system calls. In other words, the stack manager is a real-time kernel task that handles the network packets on behalf of Xenomai nucleus, sending and receiving them from the de-vice drivers that talk directly with hardware. Every data packet coming from the user space is given to the stack manager, which adds the needed headers, handles routing and manages the transmission queue for each network interface. On the other hand, when a packet is received by a network interface, an asynchronous interrupt is triggered by the network hardware device (to signal a packet reception or an error condition), the Adeos I-Pipe redirects the interrupt to Xenomai which gives control to the real-time driver. The packet is copied into memory and is passed to the stack manager, which delivers it to the user application or discards it.

5. Experimental Characterization of the wireless architecture

The described architecture has been tested in an experimental testbed made by four x86-based Personal Computers equipped with the Linux kernel 2.6.20.19. IEEE 802.11g wireless network adapters have been employed, based on the Ralink RT2500 chipset. Each card is equipped with a 2dBi omnidirectional antenna. Xenomai 2.3.4 (Gerum, 200) and Adeos I-Pipe 1.10 (Yaghmour, 2001), with RTnet version 0.9.9 (Kiszka, 2005a) have been used.

Traffic has been generated using an application written using Xenomai's Native API, which is made by several components (i.e., the *sender*, the *responder*, and the non real-time *logger*), as shown in Fig. 3.

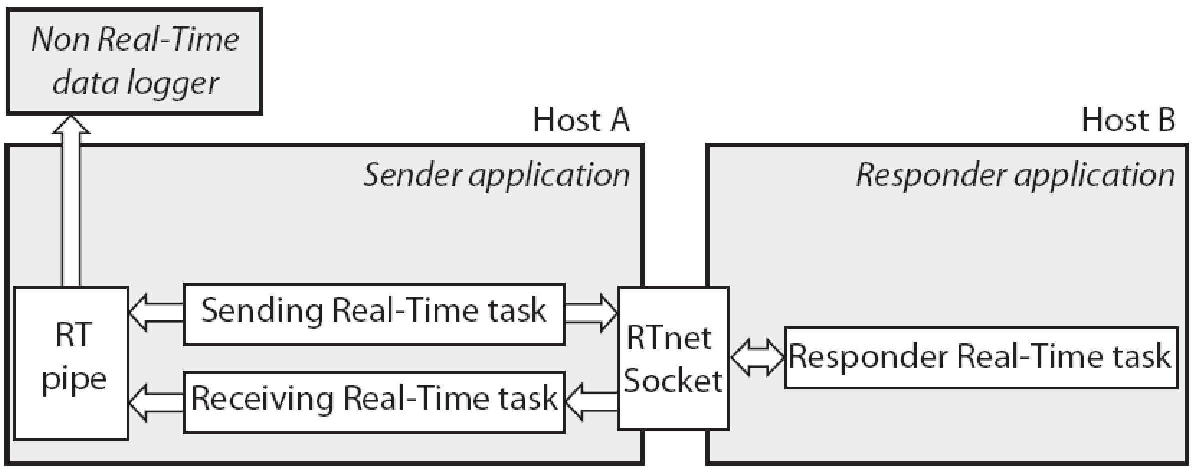


Fig. 3. Software Layout used for measurements.

The *sender* includes a real-time periodic thread which generates packets of fixed size, thus producing a Constant Bit Rate data stream. Each application packet is encapsulated in one UDP datagram. The payload contains a time-stamp measured by the application immediately before sending the packet to the lower layer.

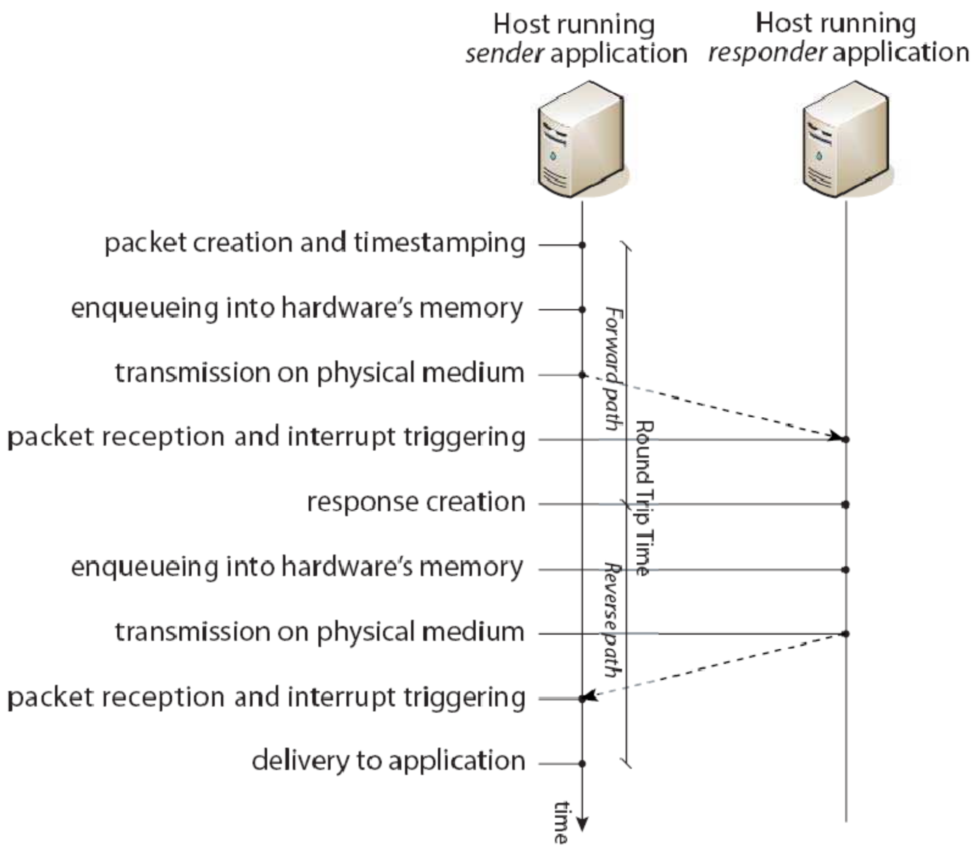

Host running *sender* application
Host running *responder* application
packet creation and timestamping
enqueueing into hardware's memory
transmission on physical medium
packet reception and interrupt triggering
response creation
enqueueing into hardware's memory
transmission on physical medium
packet reception and interrupt triggering
delivery to application
Forward path
Round Trip Time
Reverse path
time

Fig. 4. Components of Round Trip Time.

The *responder* listens for incoming packets from the *sender*; when receiving the packet, it sends back a copy of the packet, which is received by a real-time thread of the sender application. Moreover, the device drivers have been programmed to write the value of the received power (RSSI) into the packets, to constantly monitor the quality of the wireless link between the hosts.
An external, non real-time process collects from a Real-Time Pipe IPC the time values at which the real-time tasks send and receive the packets and the RSSI values, calculates the Round Trip Time (RTT) and stores the RTT and RSSI values on the disk (this approach guarantees that the I/O operations do not affect the realtime behavior of the system). The RTT includes the time elapsed by the packet to traverse the stack manager, the waiting time for the packet in the driver transmission queue before an idle channel, the transmission time, the propagation time, and the time employed by the interrupt handler and the stack manager to process the incoming packet on the receiving side, multiplied by 2 (the packet travels back and forth on the same path), as depicted in Fig. 4. Thus, the measured value is not affected by skew and offset errors, given that transmission and reception times are taken from the same clock on the sender side.
Experiments have been conducted by varying the bit rate, the background traffic load, the source transmission rate, the packet size, CW limits, interframe spaces, and the baseband processor sensitivity.
Tab. 1 summarizes all experiment parameters. The machines have been placed in a typical open space laboratory environment, and two different sets of experiments have been done.

Parameter	**1st experiment**	**2nd experiment**
DIFS [μs]	28, 10	10
S_{BP}	50-100	70
Packet Size of the Real-time Traffic [bytes]	128	128
P_{size}: Packet Size of the Background Traffic [bytes]	None	500, 1000
T_{RT} : Transmission period for Real-time traffic [ms]	10	1, 4
Transmission period for Background traffic [ms]	None	5
Bitrate [Mbit/s]	6-54	6-54
CW_{min}	31, 1	1
CW_{max}	1024, 1	1
Retry limit	0	0-2
Number of hosts	2	4

Table 1. Testbed parameters.

5.1 Experimental results without interfering traffic

In the first set of experiments, a the testbed is made by two nodes, namely A and B, placed at a close distance (around 1.5 meters), with the antennas in line of sight. The application layer of node A generates a 128 bytes long packet every 10 ms. Node B responds to packets sent by node A as shown in Fig. 3. According to Table 1, experiments are conducted for many values of S_{BP} and bit rate.

In each experiment, 10000 packets are transmitted. Each experiment has been repeated 10 times, for each couple of bit rate and S_{BP} values. In the following, in each figure, both the average value of the ten runs and the 95% confidence interval are reported. Measurements have been done turning off the ACK mechanism provided by the MAC layer.

The first set of experiments has been divided in two parts: in the first one, the wireless adapters have been configured with CW_{min} = 31, CW_{max} = 1023, and DIFS = 28 μs, which are typical values of a best-effort service. In the second part, in order to reduce the latencies due to the 802.11 MAC protocol, CW_{min} and CW_{max} have been set to 1 and the DIFS has been reduced to 10 μs, thus providing a high priority service.

Fig. 5 shows the minimum RTTs obtained for the best effort service. As expected, the minimum RTT increases when the bit rate decreases.

The most relevant contribution to the minimum RTT value is given by the time needed to access the physical medium and to transmit the data. The 802.11g standard (Walke et al., 2006) states that a wireless device must wait at least a DIFS before sending a packet, which is 28 μs long. Moreover, the transmission starts with a fixed size header (i.e., the PLCP, *Physical Layer Convergence Protocol* preamble) used for synchronization between the RF devices, which is 20 μs long (i.e., t_{PLCP} = 20μs). Then, the transmission of the frame can start. Given that the application layer payload is 128 bytes long, the whole frame is 198 bytes long (i.e., a total number of n_{bits} = 1584), adding header overhead due to UDP (8 bytes), IP (20 bytes), Ethernet encapsulation (14 bytes), and 802.11 MAC header and FCS trailer (24+4 bytes). Furthermore, a 6 μs long trailer (called Virtual Extension, VExt) is added at the end of each transmission (i.e., t_{VExt} = 6μs).

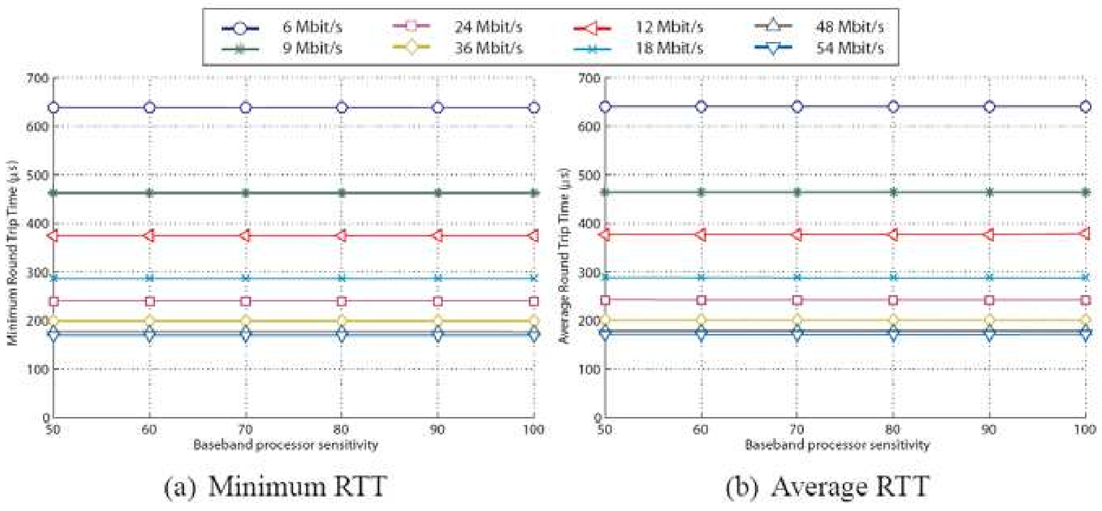


Fig. 5. Minimum and Average RTT for the best-effort service.

Thus, in absence of collisions, neglecting the very low propagation delay and the latency introduced by the real-time protocol stack, the theoretical minimum RTT is given by

$$\mathrm{RTT} = 2\,[\mathrm{DIFS} + t_{\mathrm{PLCP}} + t_{\mathrm{VExt}} + n_{\mathrm{bits}}/\mathrm{bitrate}] \quad (1)$$

which leads to the results shown in Table 2.
Such values, compared to the measured minimum values for RTT, confirm that the software latencies are almost negligible with respect to the transmission time.
Moreover, Fig. 5 shows that, for the best effort service, the average RTT is very close to the minimum one. This demonstrates that the proposed real-time communication system is able to deliver a very high quota of packets with the smallest delay.
Minimum and average RTTs obtained for the high priority service, being very similar to the one reported in Fig. 5, have not been shown.

Bitrate [Mbit/s]	**Theoretical [μs]**	**Measured [μs]**
6	636.00	638.44
9	460.00	462.50
12	372.00	374.55
18	284.00	286.48
24	240.00	240.01
36	196.00	198.53
48	174.00	176.09
54	166.66	168.93

Table 2. Minimum theoretical and measured RTTs.

The importance of a high priority service can be demonstrated by analyzing Maximum RTTs (see Fig. 6). In fact, when the best effort service is used, the maximum RTT (see Fig. 6) depends on both bit rate and baseband sensitivity. The reason is that, as S_{BP} decreases, the transmitter is more likely to detect a busy channel and then, as a consequence, latencies increase. This effect is not evident for the high priority service because shorter DIFS and CW bounds are able to reduce the impact of carrier sensing.

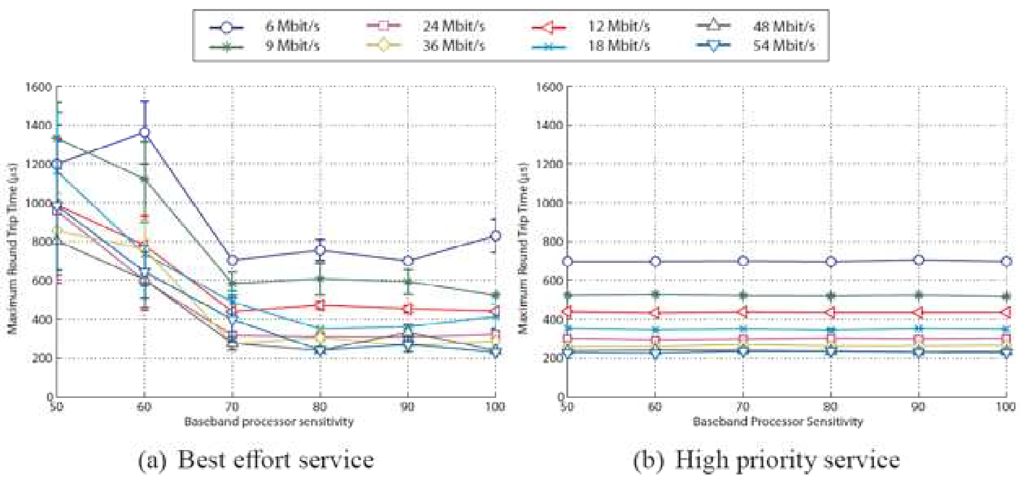


Fig. 6. Maximum RTT.

To give a further insight, Fig. 7 shows the ratio between the maximum and the minimum RTT. In the best-effort case, only when S_{BP} is greater than 70 and the bit rate is smaller than 48 Mbit/s, the ratio is smaller than 1.5. On the other hand, as expected by the previous findings, when the high priority service is considered, S_{BP} has not any influence on the maximum over minimum RTT ratio, which is always smaller than 1.4.
Fig. 8 shows the Packet Loss Ratio (PLR) computed as missing packets over total packets sent. The number of missing packets is calculated by the sender host by monitoring the number of missing response packets sent by the receiver. Note that the retransmission mechanism of the MAC layer has been turned off, so that this value actually reflects the number of packets that cannot be successfully transmitted at the first try. For both best effort and high priority services, the PLR is smaller than 0.05% when the bit rate is smaller than 48 Mbit/s; moreover, for such bit rates, in many experiments the number of lost packets equals zero. For bit rates of 48 and 54 Mbit/s a higher PLR has been measured because, if we take into account that the transmission power has been kept constant during the experiments, the higher the bitrate the weaker is the modulation scheme used to transmit with respect to noise and interferences (Walke et al., 2006).

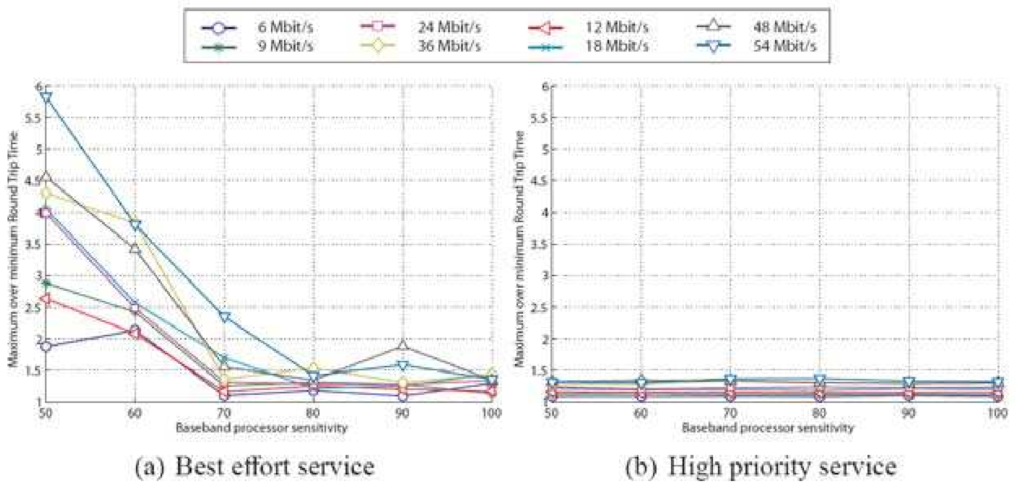
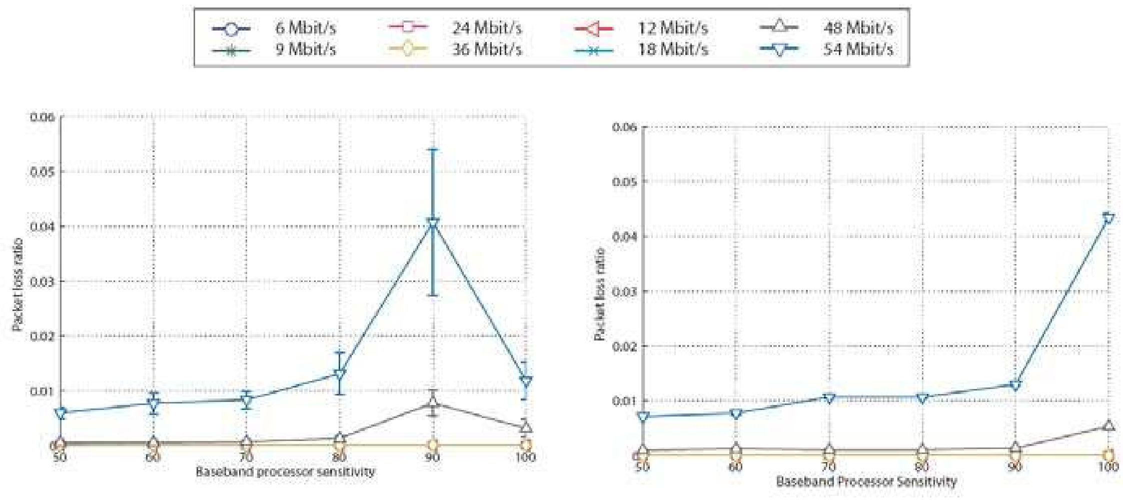


Fig. 7. Maximum over minimum RTT.

From the results presented above, we can derive as rules of thumb to use bit rates smaller or equal to 36 Mbit/s and high priority settings in order to have a robust real-time wireless communications. Anyway, these results represent optimistic performance bounds because obtained without interfering traffic. To assess their real validity, in the next sub-section we will repeat the experiments in presence of concurrent data flows when a high priority service is exploited.

5.2 Experimental results with four hosts and interfering traffic

In the second set of experiments, four machines (namely A,B,C,D) have been placed at the corners of a square area with side length of about 7 meters, at different heights and with some obstacles between them. Hosts A and B exchange real-time data, whereas C and D generate interfering traffic. In particular, using the software described in Fig. 3, A and C generate a CBR traffic sent to B and D, respectively.
For what concerns hosts A and B, interfaces are set in order to provide a high priority service. With respect to the previous experiments, the impact of 802.11 frame retransmission mechanism has been also considered. For that purpose, experiments have been conducted with 802.11 acknowledgment turned off and on, respectively. When turned on, the maximum number of retransmission for each frame has been varied from 1 to 2.
Regarding hosts C and D, default settings for 802.11 interfaces have been used, i.e., CW_{min}=31, CW_{max}=1023, DIFS equal to 28μs, bitrate equal to 54 Mbit/s, acknowledgment mechanism turned on with 7 maximum retries.
According to Table 1, for real-time traffic, 128 bytes long packets are transmitted with inter-transmission times (T_{RT}) equal to 1 ms or 4 ms, whereas for the interfering traffic, the inter-transmission time is 5 ms. By taking into account retransmissions, we have measured an actual inter-transmission time of 1 ms at MAC layer, which is comparable with the inter-transmission time of the real-time traffic. Thus, a stress test has been considered.



(a) Best effort service (b) High priority service

Fig. 8. Packet Loss Rate.

Two different sizes for the packet size, P_{size}, of the background traffic have been used (500 and 1000 bytes), in order to evaluate the impact of such factor on the performance of the system.

Lastly, we have only considered a S_{BP} value equal to 70, which provided the best performance in the previous set of experiments.
The minimum measured values for RTT are very similar to those already shown in Fig. 5, thus they have not been reported again. The maximum RTT is shown in Fig. 9.
Obviously, for T_{RT} equal to 1 ms we obtain a larger maximum RTT with respect to the case of T_{RT} equal to 4 ms. The reason is that if the transmission rate of real-time packets increases, the probability of finding the channel busy grows too. It is worth to note that, when the inter-transmission time is equal to 4 ms, the RTT is never bigger than 4 ms, which means that transmissions of a packet never overlap with subsequent transmissions. The same conclusion does not hold when the inter-transmission time is 1 ms, which implies that the delay experienced by a real-time packet can influence the delay of future packets. As a consequence, the measured value for the maximum becomes extremely high, about 16 ms in the worst case.
A bitrate of 6 Mbit/s gives the worst performance due to the largest transmission time. Anyway, only a minimal fraction of the frames sent is affected by such high delays. As depicted in Fig. 10, the higher bound on the RTT for 98% of the successfully received frames is much smaller than the maximum measured RTT. Higher bitrates (48 and 54 Mbit/s) already showed to be less robust in the previous experiment, leading to higher values for the overall delay, while a bit rate of 6 Mbit/s needs too much time for the frames to be transmitted without collisions. Intermediate values of the bit rate (9 to 36 Mbit/s) give a RTT that is almost smaller than 2 ms. Thus, with a good degree of approximation we can state that the one-way delay will be no more than a millisecond for such packets.

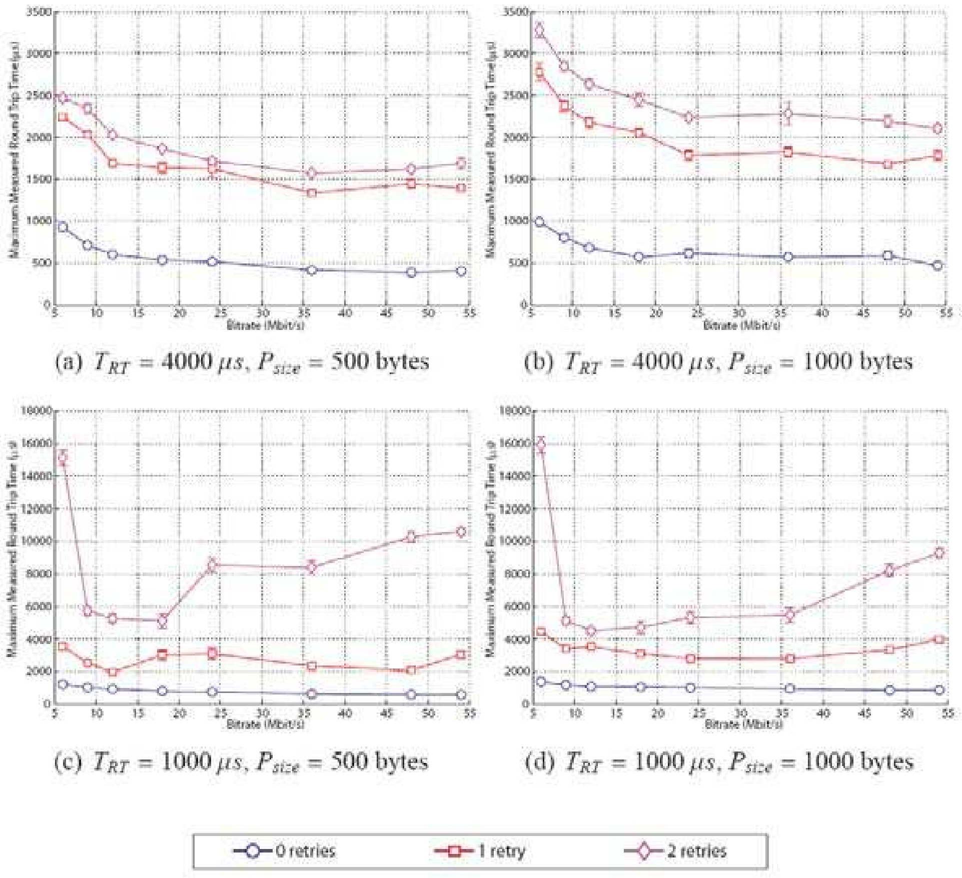
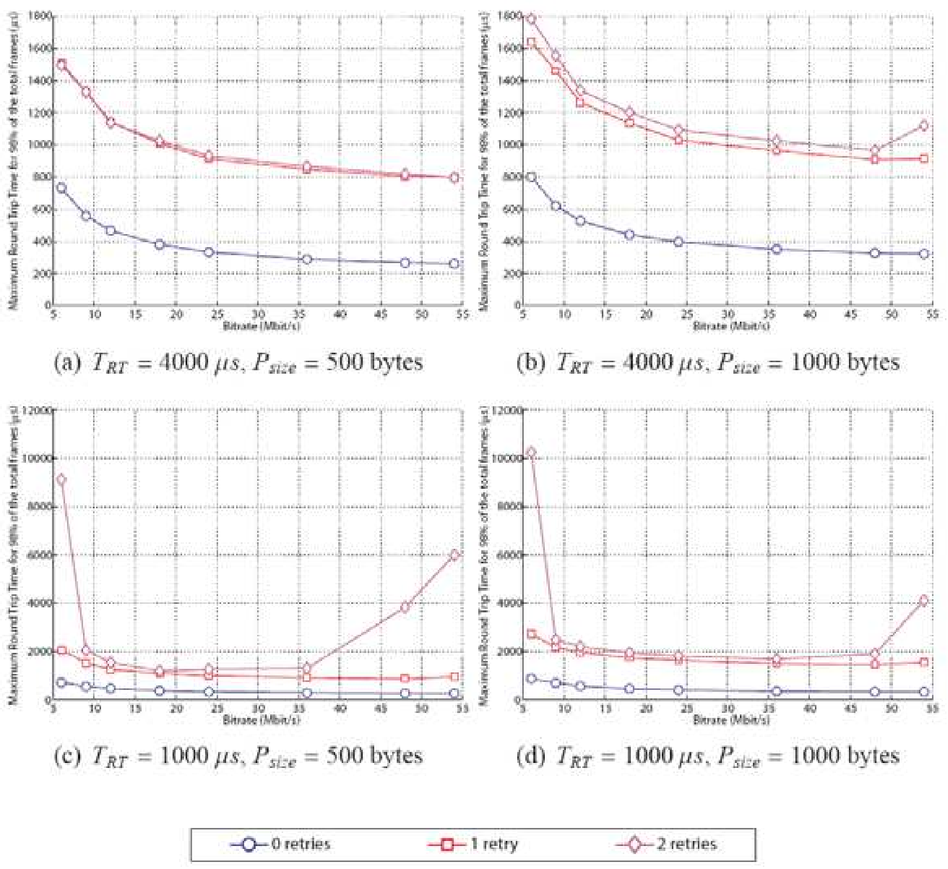


Fig. 9. Maximum measured Round Trip Time.

The average RTT is depicted in Fig. 11. It is a decreasing function of the bitrate up to 36 Mbit/s (again, higher the bitrate is and higher the collision probability is). If the retransmissions are enabled, the average value slightly increases, due to the transmission of the acknowledgment frame. In fact, after the successful reception of a frame the host must wait for a SIFS before sending the acknowledgment frame. Thus, the RTT increases as another packet must be sent.



(a) $T_{RT} = 4000\,\mu s$, $P_{size} = 500$ bytes

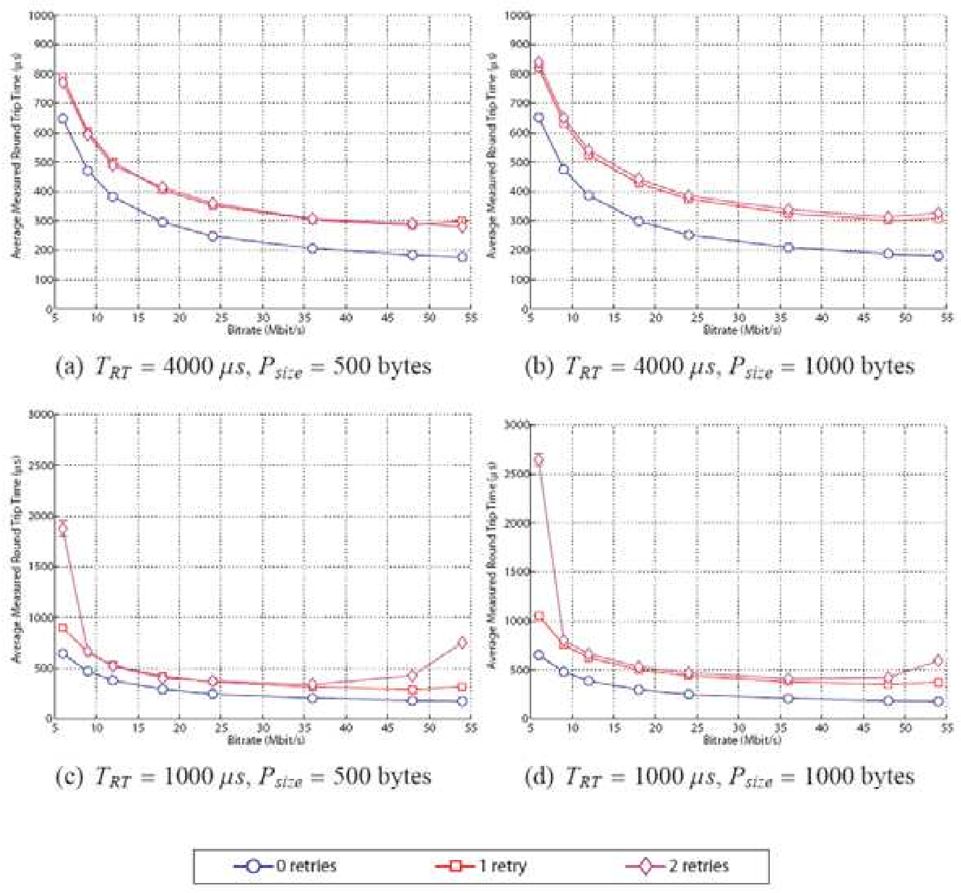
(b) $T_{RT} = 4000\,\mu s$, $P_{size} = 1000$ bytes

(c) $T_{RT} = 1000\,\mu s$, $P_{size} = 500$ bytes

(d) $T_{RT} = 1000\,\mu s$, $P_{size} = 1000$ bytes

Fig. 10. Maximum measured Round Trip Time (98% of the received frames).

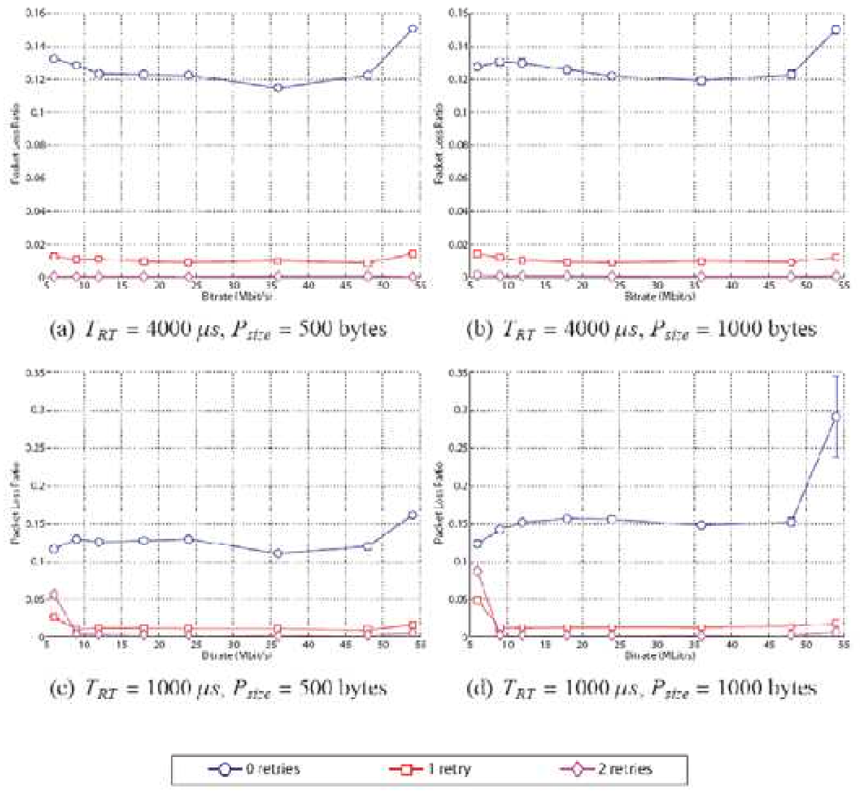
As expected, the performance of the system in terms of packet loss is greatly influenced by the presence of interfering traffic on the channel (see Fig. 12). Without a retransmission mechanism, packet losses are unacceptable, ranging from 12% to 15%. Apart from the 54 Mbit/s bitrate value, the packet loss rate is not directly related to the bitrate value. The retransmission mechanism reduces the PLR to values smaller than 1%, except for the 6 Mbit/s case, where the losses are high even with retrasmissions enabled. Thus, bitrates between 12 Mbit/s and 36 Mbit/s seem to be the best tradeoff between packet loss rate and maximum delay, showing a robust and reliable behavior in all the considered scenarios.
The empirical Cumulative Distribution Functions of the RTT for a bitrate of 36 Mbit/s (which has given the best tradeoff between PLR and delay) is depicted in Fig. 13. If we do not use the retransmission mechanism the knee of the function is very close to the minimum value for the RTT, but the graph has been traced only considering the successfully transmitted packets, so it does take into account the heavy PLR. With retransmissions enabled, the system guarantees that at least 75% of the packet exchanged can be delivered in a timely manner, that is, with a value which is quite close to the minimum measured.



(a) T_{RT} = 4000 μs, P_{size} = 500 bytes (b) T_{RT} = 4000 μs, P_{size} = 1000 bytes

(c) T_{RT} = 1000 μs, P_{size} = 500 bytes (d) T_{RT} = 1000 μs, P_{size} = 1000 bytes

Fig. 11. Average measured Round Trip Time.



(a) T_{RT} = 4000 μs, P_{size} = 500 bytes (b) T_{RT} = 4000 μs, P_{size} = 1000 bytes

(c) T_{RT} = 1000 μs, P_{size} = 500 bytes (d) T_{RT} = 1000 μs, P_{size} = 1000 bytes

Fig. 12. Packet Loss Rate.

To conclude, experiments clearly show that the considered communication architecture, with a proper setting of networking interface card parameters, provides very tight bounds on both packet delays and losses. In particular, using high priority settings, with bitrates ranging from 12 -36 Mbit/s and retry limit equal to 2, the 98% of packets experience a RTT no larger than 2 ms with negligible packet losses, even in the presence of interferring traffic. These results demonstrate that a further step toward Wireless Networked Control Systems has been done.

6. Proposed TDMA scheme for a Wireless Networked Control System

The RTnet TDMA access discipline has been conceived for wired networks, which are much more reliable and less subject to electromagnetic interferences than wireless ones. As a consequence, the expected delay jitter of wireless networks can make troublesome its exact calibration.
In fact, the carrier sense mechanism of 802.11 may delay the transmission of a packet, making it overlap with subsequent timeslots and causing a chain effect which could break the whole system synchronisation. Lastly, the transmission of outgoing packets on each station is triggered by the reception of the synchronisation packets; a station that does not receive such synchronisation packets loses the chance to transmit into the current TDMA frame.

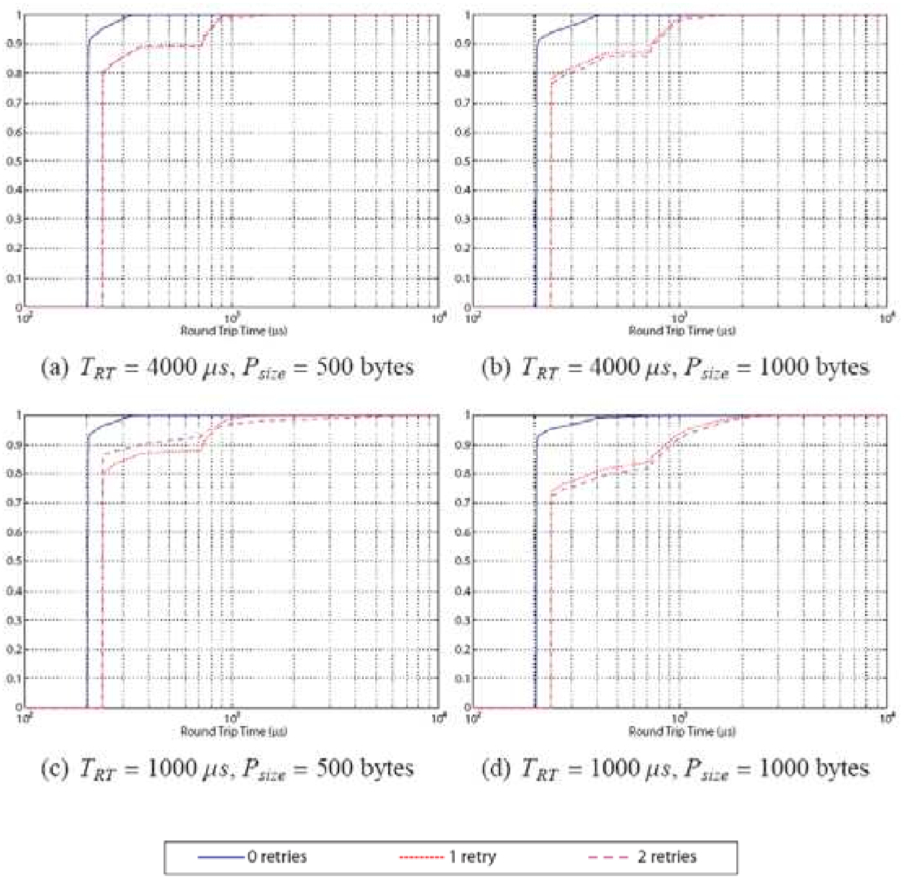


Fig. 13. Cumulative distribution functions of measured RTT.

This can lead to instability, as the output queues may fill up causing unacceptable delays or packet losses. In order to make the current TDMA implementation work reliably over 802.11 networks, we hereby propose a few modifications that could be applied to cope with the characteristics of the wireless channel:

- **Protection mechanisms**: to reduce the probability of packet collisions, a protection scheme should be used. The IEEE 802.11g amendment provides a CTS-to-self mechanism that a station can use to assign to itself the right to access the channel, as it received a RTS (Request to Send) from another station (Walke et al., 2006).
- **Ordered retransmission scheme:** the retransmission mechanism used in 802.11 tries to retransmit the lost packet as soon as the channel is sensed idle again. This approach can not be safely used with a TDMA scheme, as the packet retransmission may overlap with a timeslot assigned to another station. It is necessary to implement a retransmission scheme able to defer the retransmission only when all the stations have already transmitted their data.
- **Automatic recovery of timing**: when a synchronisation packet is not correctly received, a station should be able to synchronize automatically with the start of the TDMA frame, in order not to lose its chance to transmit during the current TDMA frame. This can be safely done for short periods of time using a proper weighted average of the interarrival times of the last synchronisation frames, if a station does not miss more than a few synchronisation rounds consecutively. Otherwise the start of TDMA frame may not be calculated correctly anymore due to clock drift. The use of backup masters can reduce the occurrences of such event.
- **Active monitoring**: a station wishing to transmit may choose to actively monitor the channel activity before sending the packet to the hardware. If the channel is sensed busy for too much time, the driver may decide to drop the packet instead of sending it, in order to avoid overlapping with other timeslots.

These solutions should be easily implemented on generic 802.11 network adapters, as long as the source code for the device driver is available and the firmware lets the user control the basic MAC parameters. The proposed approach should make the real-time stream more robust with respect to interferences. The actual performance increase has still to be evaluated. At the moment, we have implemented an ordered retransmission scheme which defers the retransmission of a packet to a special timeslot at the end of the cycle. We have considered the TDMA frame structure made of two parts, as described in Fig. 14.

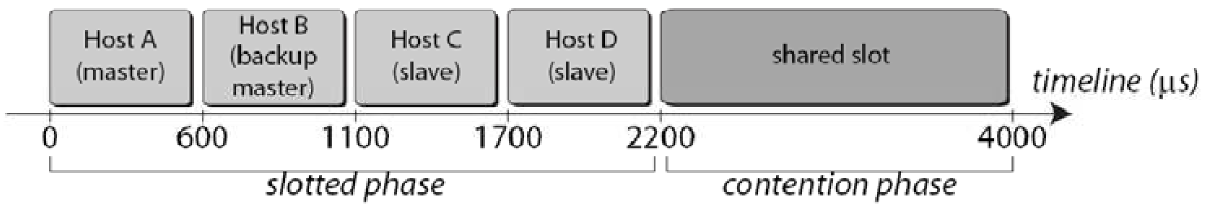


Fig. 14. Proposed TDMA frame structure.

The first part, called *slotted phase*, contains normal TDMA timeslots, which can be assigned to each station. The second part, called *contention phase*, contains a single timeslot, which is shared by all the stations for retransmissions. For the time being, a single retransmission attempt for each frame has been considered.The 802.11 acknowledgment mechanism can be used or the packets can be acknowledged at application layer. Thus, any packet that has not been acknowledged at the end of the slotted phase can be sent during the contention phase.

7. Experimental Analysis of the proposed TDMA scheme

The real-time communications architecture used to test the TDMA scheme is the same one considered in Sec. 5. Each machine has been equipped with a Ralink RT2500 WLAN adapter. The host have been placed in a laboratory open space, at a distance of approximatively 10 meters among each other. A simple TDMA network has been created among the hosts, one of which has acted as master, another as backup master, and the others as slaves, as reported in Fig. 14. The DIFS value has been set to 10 µs, and the CW_{min} and CW_{max} values have been set to their standard minimum value (Walke et al., 2006). Finally, the S_{BP} value has been set to 80, which is a good tradeoff between latencies and packet loss rate, as shown in Sec. 5. These settings guarantee the highest priority on the wireless channel. We have considered all bit rate values allowed by 802.11g amendment for OFDM operation, from a minimum of 6 Mbit/s to a maximum of 54 Mbit/s Walke et al., 2006). A wireless channel almost free from other transmissions has been considered during the measurements. The IEEE 802.11 retransmission mechanism has been turned off.

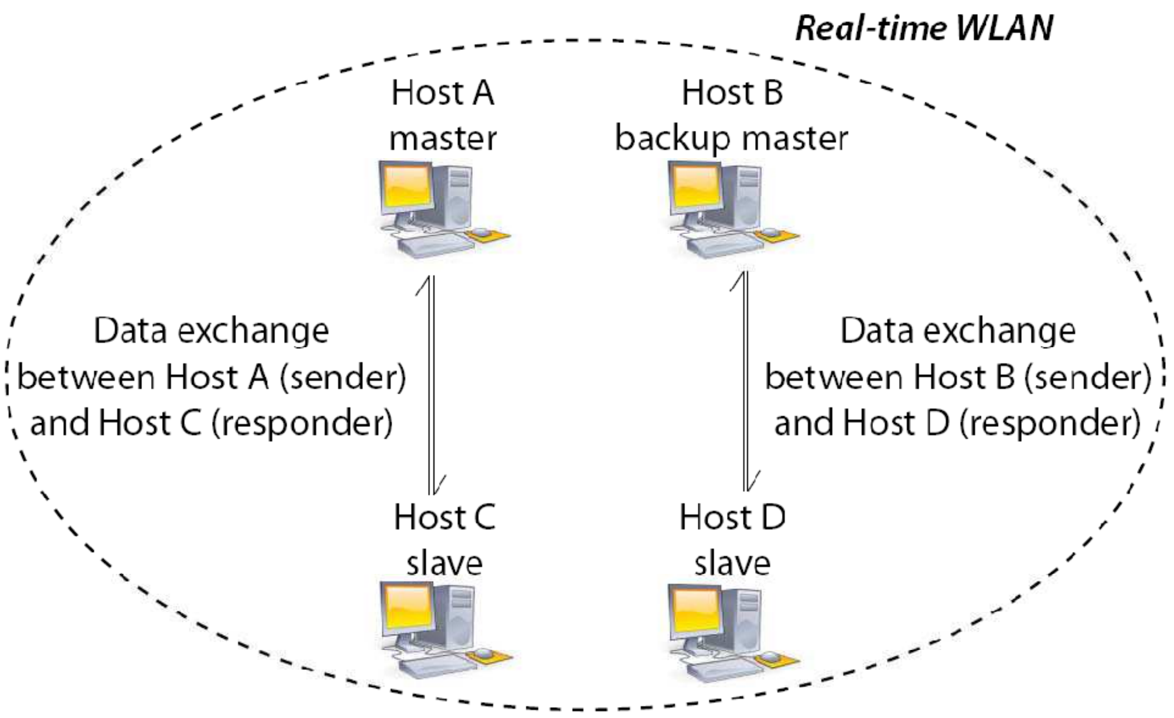


Fig. 15. Testbed used for measurements.

Traffic has been generated using a custom application made by several components (i.e., the *sender*, the *responder*, and a non real-time data logger). The *sender* includes a periodic thread which generates 128 bytes long packets, synchronised with the TDMA frame (i.e., 4 ms), so that it generates a new packet for every cycle; packets at application layer are encapsulated in UDP datagrams. The sync reception acts as a trigger for packet creation. The packet contains a unique identifier and a time-stamp taken by the application when the packet is created. The *responder* listens for incoming packets from the sender; when receiving the packet, it sends back a copy of the packet, which is received by a thread of the sender application. An external process collects from a Real-Time Pipe IPC the transmission and reception time values, calculates the Round Trip Time (RTT) and stores the values on disk. The *sender* has been run on hosts A and B, sending traffic to hosts C and D respectively, which were running the *responder* software, as depicted in Fig. 15.

When the *sender* host does not receive an answer during the slotted phase, it retransmits the packet in the contention phase, and so does the *responder*. Moreover, if the measured RTT of a packet is greater than the TDMA frame length, the responder host may have lost a TDMA synchronization packet. Thus, a response packet is still held into the output queue of the responder host. When this happens, the *sender* notifies the *responder* which uses the contention phase to send the packet left in its queue. In each experiment the hosts exchanged 5000 packets, and each experiment has been repeated 5 times, for a total of 25000 packets exchanged for each bit rate value considered.

Fig. 16 shows the minimum and the maximum RTT measured for both real-time streams. Given that hosts A and B take a new timestamp at the beginning of each frame, the minimum RTTs are very close to the timeslot offsets of the responder hosts C and D, respectively. Moreover, they decrease with the bit rate due to a smaller transmission time. The maximum RTT is mostly due to the missed reception of the sync frame by the slave nodes. The obtained values indicate that the retransmission scheme is able to recover the normal system behavior in at most two TDMA frames.

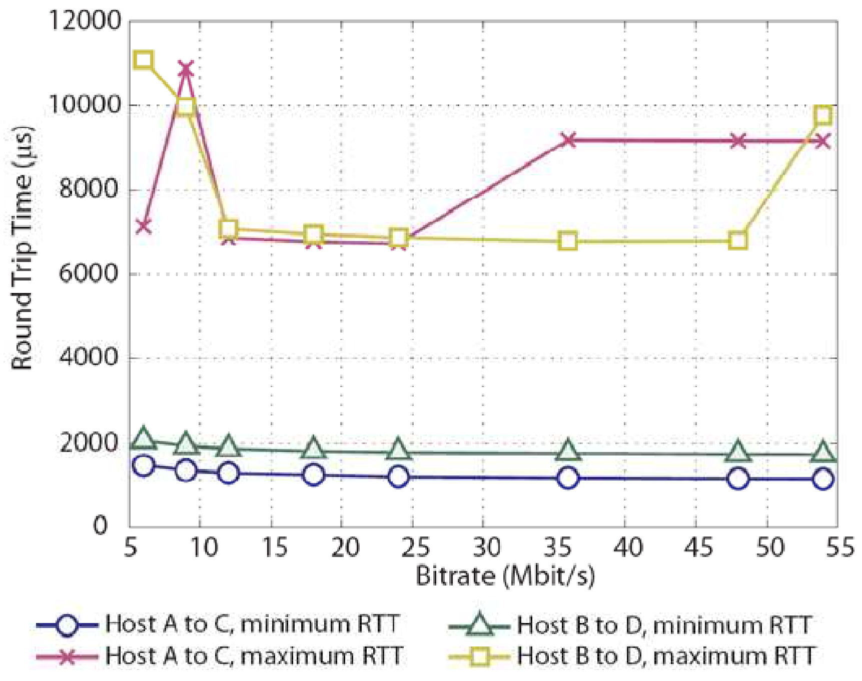


Fig. 16. Minimum and Maximum RTT.

Fig. 17 shows the percentage of lost and recovered packets for each considered bit rate. Most of the packets have been recovered by the retransmission scheme, though it allows just one retransmission. Losses are very small, always less than 0.3% of the total sent packets.

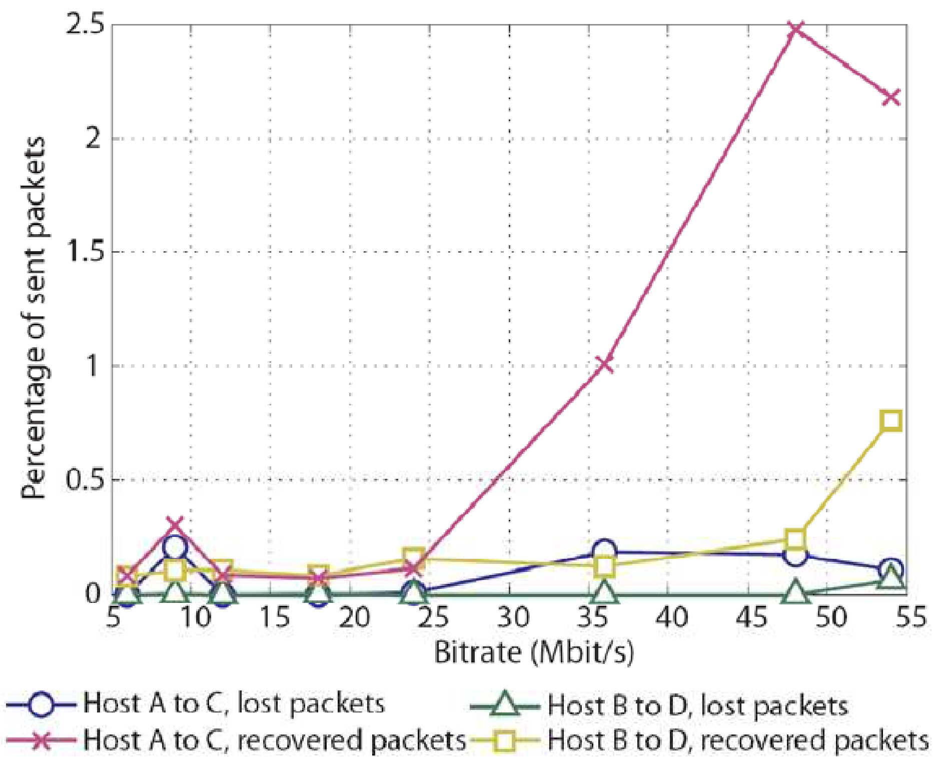


Fig. 17. Percentage of lost and recovered packets.

8. Conclusions

This chapter has analyzed the feasibility of a 802.11 based wireless real-time communication system. For that purpose a wireless communication architecture that properly integrates the leading IEEE 802.11 technology, the RTnet framework, and the Xenomay nano-kernel has been implemented. This architecture has been experimentally tested for various transmission data rates, baseband processor sensitivities, and sampling intervals, with and without interfering traffic. Experiments have demonstrated that by properly setting protocol parameters a robust real-time service can be provided.

9. References

Andersson M.; Henriksson D.; Cervin A. & Arzen K. E. (2005). Simulation of wireless networked control systems, *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, Seville, Spain, Dec. 2005.

ANSI (1994). *Portable Operating Sytem Interface (Posix)*. ANSI, IEEE, 1994.

Baillieul J. & Antsaklis P. J. (2007). Control and Communication Challenges in Networked Real-time Systems. *Proceedings of the IEEE*, Vol. 95, No. 1, (Jan. 2007) 9-28.

Baliga G. & Kumar P. R. (2005). A middleware for control over networks, *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, Seville, Spain, Dec. 2005.

Barr M. (2003). Choosing an RTOS. *Tech. Rep. Embedded Systems Programming*, Jan. 2003.

Benvenuti C. (2006). *Understanding Linux Network Internals*. O'Reilly, 2006.

Biasi M. D.; Snickars C.; Landernas K. & Isaksson A. J. (2008). Simulation of process control with wirelesshart networks subject to packet losses, *Proceedings of 4th IEEE Conference on Automation Science and Engineering*, Washington DC, USA, Aug. 2008.

Boggia G.; Camarda P.; Grieco L. A. & Zacheo G. (2008a). Toward wireless networked control systems: an experimental study on real-time communications in 802.11 wlans, *Proceedings of 7th IEEE International Workshop on Factory Communication Systems, WFCS*, Dresden, Germany, May 2008.

Boggia G.; Camarda P.; Grieco L. A. & Zacheo G. (2008b). An experimental evaluation on using TDMA over 802.11 MAC for wireless networked control, *Proceedings of Emerging Technologies and Factory Automation, ETFA*, Hamburg, Germany, Sep. 2008.

Boughanmi N.; Song Y. & Rondeau E. (2008). Wireless networked control system using IEEE 802.15.4 with GTS, *Proceedings of 2nd Junior Researcher Workshop on Real-Time Computing, JRWRTC*, Rennes, Brittany, Oct. 2008.

Burda R. & Wietfeld C. (2007). Multimedia over 802.15.4 and ZigBee Networks for Ambient Environmental Control, Proceedings of the IEEE VTC Spring, Dublin, Ireland, Apr. 2007.

Buttazzo G.; Velasco M. & Martì P. (2007). Quality-of-Control Management in Overloaded Real-time Systems. *IEEE Trans. on Computers*, Vol. 56, No. 2, (Feb. 2007) 253–266.

Cena G.; Bertolotti I. C.; Valenzano A. & Zunino C. (2007). Evaluation of response times in industrial WLANs. *IEEE Trans. on Industrial Informatics*, Vol. 3, No. 3, (Aug. 2007) 191–201.

Cervin A.; Ohlin M. & Henriksson D. (2007). Simulation of networked control systems using truetime, *Proceedings of the 3rd International Workshop on Networked Control Systems: Tolerant to Faults*, Nancy, France, Jun. 2007.

Chen J.; McKernan A.; Irwin G. W. & Scanlon W. G. (2008). Experimental characterisation and analysis of wireless network control systems, *Proceedings of the IET Irish Signals and Systems Conference, ISSC*, Galway, Ireland, Jun. 2008.

Choi D. H.; Lee J. I.; Kim D. S. & Park W. C. (2006). Design and implementation of wireless fieldbus for networked control systems, *Proceedings of SICE-ICASE International Joint Conference*, Bexco, Busan, Korea, Oct. 2006.

DIAPM (2008), Real-time application interface (RTAI) for Linux. *Tech Rep. Politecnico di Milano*, 2008, available online: http://www.rtai.org.

Flammini A.; Marioli D.; Sisinni E. & Taroni A. (2009). Design and implementation of a wireless fieldbus for plastic machineries. *IEEE Trans. on Industrial Electronics*, Vol. 56, No. 3, (Mar. 2009) 747–755.

Floroiu J.; Ionescu T. C.; Ruppelt R.; Henckel B. & Mateescu M. (2001). Using NDIS intermediate drivers for extending the protocol stack. a case-study. *Computer Communications*, Vol. 24, No. 7-8, (Apr. 2001) 703–715.

Gerum P. (2004). Xenomai – implementing a RTOS emulation framework on GNU/Linux. Available online : http://www.xenomai.org/documentation.

Hasan M. S.; Yu H.; Griffiths A.& Yang T. C. (2007). Simulation of distributed wireless networked control systems over MANET using OPNET, *Proceedings of the IEEE International Conference on Networking, Sensing and Control*, London, UK, Apr. 2007.

Hespanha J. P.; Naghshtabrizi P. & Xu Y. (2007). A Survey of Recent Results in Networked Control Systems. *Proceedings of the IEEE*, Vol. 95, No. 1, (Jan. 2007) 138–162.

Heynicke R.; Kruger D.; Wattar H. & Scholl G. (2008). Modular wireless fieldbus gateway for fast and reliable sensor/actuator communication, *Proceedings of Emerging Technologies and Factory Automation, ETFA*, Hamburg, Germany, Sep. 2008.

IEEE (1999a). *IEEE 802.11, Information Technology -Telecommunications and Information Exchange between Systems. Local and Metropolitan Area Networks. Specific Requirements. Part 11: Wireless LAN MAC and PHY Specifications, 1st ed., ANSI/IEEE Std. 802.11, ISO/IEC 8802-11.* IEEE standard for Information Technology, 1999.

IEEE (1999b). *IEEE 802.11, Supplement to IEEE Standard for Information Technology. Local and Metropolitan Area Networks. Specific Requirements. Part 11: Wireless LAN MAC and PHY Specifications: Higher-Speed Physical Layer Extension in the 5 GHz Band, IEEE Std 802.11a, ISO/IEC 8802-11:1999/Amd 1:2000(E).* IEEE standard for Information Technology, 1999.

IEEE (1999c). *Supplement to IEEE Standard for Information Technology. Local and Metropolitan Area Networks. Specific Requirements. Part 11: Wireless LAN MAC and PHY Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band, IEEE Std 802.11b).* IEEE standard for Information Technology, 1999.

IEEE (2003). *Supplement to IEEE Standard for Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band, IEEE Std 802.11g.* IEEE standard for Information Technology, 2003.

IEEE (2005). *Amendment to Standard for Information Technology. LAN/MAN Specific Requirements -Part 11: Wireless MAC and PHY Specifications: MAC Quality of Service (QoS) Enhancements, IEEE 802.11e/D13.0.* IEEE standard for Information Technology, 2005.

IEEE (2006). *Std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* IEEE standard for Information Technology, Sept. 2006.

Kim W.; Ji K. & Ambike A. (2006). Real-time Operating Environment for Networked Control Systems. *IEEE Trans. on Automation Science and Engineering*, Vol. 3, No. 3, (Jul. 2006) 287–296.

Kiszka J. (2005b). The real-time driver model and first applications. *Tech. Rep. Xenomai*, available online: http://www.xenomai.org/documentation/

Kiszka J.; Wagner B.; Zhang Y. & Broenink J. (2005a). RTnet – a flexible hard real-time networking framework, *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation*, Catania, Italy, Sep. 2005.

Krber H.J.; Wattar H. & Scholl G. (2007). Modular wireless real-time sensor/actuator network for factory automation applications. *IEEE Trans. on Industrial Informatics*, Vol. 3, No. 2, (May 2007) 111–119.

Lee S.; Park J. H.; Ha K. N. & Lee K. C. (2008). Wireless networked control system using NDIS-based four-layer architecture for IEEE 802.11b, *Proceedings of IEEE Int. Workshop on Factory Communication Systems, WFCS*), Dresden, Germany, May 2008.

Liu G. P.; Xia Y.; Chen J.; Rees D. & Hu W. (2007). Networked Predictive Control of Systems with Random Network Delays in both Forward and Feedback Channels. *IEEE Trans. on Industrial Electronics*, Vol. 54, No. 3, (Jun. 2007) 1282–1297.

Massa A. (2002). *Embedded Software Development with ECos.* Prentice Hall PTR, 2002.

McKenney P. (2005). A realtime preemption overview. *LWN.net*, available online: http://lwn.net/Articles/146861.

Moyne J. R. & Tilbury D. M. (2007). The Emergence of Industrial Control Networks for Manufactoring Control, Diagnostics, and Safety Data. *Proceedings of the IEEE*, Vol. 95, No. 1, (Jan. 2007) 29–47.

Nair G. N.; Fagnani F.; Zampieri S. & Evans R. J. (2007). Feedback Control Under Data Rate Constraints: An Overview. *Proceedings of the IEEE*, Vol. 95, No. 1, (Jan. 2007) 108–137.

Nethi S. ; Pohjola M.; Eriksson L. & Jntti R. (2007). Platform for emulating networked control systems in laboratory environments, *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM)*, Helsinki, Finland, Jun. 2007.

Neumann P. (2007). Communication in industrial automation what is going on?. *Control Engineering Practice*, Vol. 15, No. 11 (Nov. 2007) 1332-1347.

Pellegrini F. D.; Miorandi D.; Vitturi S. & A. Zanella (2006). On the use of wireless networks at low level of factory automation systems. *IEEE Trans. on Industrial Informatics*, Vol. 2, No. 2, (May 2006) 129–143.

Ralink (2006) Technologies. Ralink rt2500 chipset overview. *Tech Rep. Ralink Technologies*, available online: http://www.ralinktech.com

Rauchhaupt l. (2002). System and device architecture of a radio based fieldbusthe rfieldbus system, *Proceedings of the 4th IEEE International Workshop on Factory Communication Systems*, Vasteras, Sweden, Aug. 2002.

Robinson C. L. & Kumar P. R. (2007). Sending the most recent observation is not optimal in networked control: Linear temporal coding and towards the design of a control specific transport protocol, *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, Louisiana, USA, Dec. 2007.

Schenato L.; Sinopoli B.; Franceschetti M.; Poolla K. & Sastry S. S. (2007). Foundations of Control and Estimation over Lossy Networks. *Proceedings of the IEEE*, Vol. 95, No. 1, (Jan. 2007) 163–187.

Silberschatz A.; Galvin P. B. & Gagne G. *Operating System Concepts (7th Edition)*. Wiley, 2004.

Song J.; Han S.; Mok A. K.; Chen D.; Lucas M.; Nixon M. & Pratt W. (2008). Wirelesshart: Applying wireless technology in real-time industrial process control, *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium*, St. Louis, MO, USA, Apr. 2008.

Straumann T. (2001). Open source real time operating systems overview, *Proceedings of the 8th International Conference on Accelerator and Large Experimental Physics Control Systems*, San Jose, CA, USA, 2001.

Tabbara M.; Nesic D. & Teel A. R. (2007). Stability of Wireless and Wireline Networked Control Systems. *IEEE Trans. on Automatic Control*, Vol. 52, No. 9, (Sep. 2007) 1615–1630.

Varshney U. (2003). The Status and Future of 802.11-Based WLANs. *IEEE Computer*, Vol. 36, No. 6, (Jun. 2003) 102–105.

Walke B. H.; Mangold S. & Berlemann L. (2006). *IEEE 802 Wireless Systems*, John Wiley and Sons, 2006.

Willig A.; Matheus K. & Wolisz A. (2005). Wireless technologies in industrial networks. *Proceedings of IEEE*, Vol. 93, No. 6, (Jun. 2005) 1130–1150.

Wu J. & Chen T. (2007). Design of Networked Control Systems with Packet Dropouts," *IEEE Trans. on Automatic Control*, Vol. 52, No. 7, (Jul. 2007) 1314–1319.

Yaghmour K. (2001). Adaptive domain environment for operating systems. *Tech. Report Adeos*, available online: http://www.opersys.com/ftp/pub/Adeos/adeos.pdf.