

A Reprogrammable and Intelligent Monitoring System for Rock-Collapse Forecasting

Cesare Alippi, *Fellow, IEEE*, Giacomo Boracchi, and Manuel Roveri

Abstract—The lack of clearly noticeable forerunners, the need to acquire large amounts of data coming from sensors sampled at middle/high rates, and the potentially catastrophic effects of the physical phenomenon under monitoring make rock-collapse forecasting a challenging and valuable environmental monitoring application. In this paper, we describe a rock-collapse forecasting system based on a hybrid wireless–wired architecture, where a set of acquisition units is connected through a fieldbus to a base station, which collects and wirelessly transmits acquired measurements to a remote control room. The main features of the proposed rock-collapse forecasting system are the ability to process data sampled at high-frequency rates locally and in real time, the run-time remote reconfigurability of the forecasting application, and the possibility to distribute intelligent processing through the system layers to balance energy consumption and the application performance. Five instances of the proposed system have been deployed along the Swiss–Italian Alps.

Index Terms—Distributed monitoring systems, high-frequency-sampling rock-collapse forecasting, hybrid sensor networks, intelligent monitoring systems, remote reconfigurability.

I. INTRODUCTION

AMONG the wide range of physical phenomena threatening mountain regions, the collapse of rock faces represents one of the most dangerous and is hard to predict. In fact, rock collapse is characterized by a rapid dynamic just before the fall and might induce catastrophic effects whenever human settlements, roads, or critical infrastructures are affected. For these reasons, an effective and real-time monitoring and forecasting action is required to promptly raise alarms and possibly activate emergency procedures.

Environmental forces such as those induced by thermal stress, frost/defrost cycles during winter, and gravitation significantly contribute to the fall, but their role is only partly understood. From a geophysical perspective, rock collapse is the final phase of a continuous process induced by the coalescence of *microfractures* into larger fractures. Interestingly, during this coalescence phase, microacoustic emissions, i.e., burst signals resembling in shape those generated by earthquakes, are emitted whenever a fracture enlarges its size. These bursts represent extra information that a suitable monitoring system can acquire to open further views on the physical phenomenon under observation. Microacoustic emissions are characterized by middle/high sampling rates; we bounded the sampling frequency to 2 kHz, which is a frequency coarsely associated

with evolving millimetric-size fractures. The choice implies that we observe burst signals with a spectrum of up to 1 kHz, which is a reasonable frequency according to geophysicists. This design constraint is appropriate since it is known that frequencies progressively reduce when getting closer to the rock fall [1].

The proposed rock-collapse forecasting system is based on a hybrid wireless–wired hardware solution for structural health and environmental monitoring applications presented in [2]. The peculiar characteristic of the hardware is the combined use of wireless and wired communication, i.e., sensing units are connected through a fieldbus to a base station (BS), which wirelessly transmits the acquired data (or the features/events extracted from them) to a control room. By relying on such a hardware architecture, we developed a novel rock-collapse forecasting system whose distinctive features are as follows.

- 1) The ability to *acquire and locally process* multiple scalar signals with a sampling rate of up to 2 kHz.
- 2) The presence of *intelligent processing* distributed through the network aiming at reducing the overall energy consumption while maintaining the application performance. The first level consists in a local trigger that analyzes online the acquired data to identify those events that are worthy to be remotely transmitted. Another intelligent mechanism autonomously switches the working modality of the monitoring system between a normal mode (periodic transmission) and an alarm mode (a potentially critical situation). Finally, the intelligence is considered at the event-classification level in the control room where false alarms are automatically discarded.
- 3) The *reprogrammability* of the application at run time, i.e., the ability to modify or update the software code and/or the parameters of the sensing units.
- 4) A two-layer synchronization mechanism for hybrid wireless–wired sensor networks guaranteeing strict synchronization among the acquisition units (less than 1 ms).

We emphasize that we currently have five operational deployments of the proposed system for rock-collapse forecasting in Northern Italy and Switzerland; the details of these deployments are given in Table I. Since the analysis of microacoustic emissions for rock-collapse forecasting is still a novel and challenging research area, the design and development of these systems required a strict collaboration with the experts of the field (i.e., geologists and geophysicists). Remarkably, the data acquired by our systems represent valuable information both from the geological/geophysical point of view (to improve the geological/geophysical knowledge about the collapse of rock faces) and from the technological point of view (to increase the effectiveness and robustness of our systems).

Manuscript received August 29, 2013; revised February 20, 2014, July 21, 2014, October 31, 2014, and February 10, 2015; accepted April 30, 2015.

The authors are with Dipartimento di Elettronica e Informazione, Politecnico di Milano, 20133 Milan, Italy (e-mail: cesare.alippi@polimi.it; giacomo.boracchi@polimi.it; manuel.roveri@polimi.it).

Digital Object Identifier 10.1109/JSYST.2015.2429913

TABLE I
DEPLOYMENTS OF THE PROPOSED SYSTEM FOR ROCK-COLLAPSE
FORECASTING IN NORTHERN ITALY AND SWITZERLAND

Deployment	Geographical area	Active since	Number of Units
St. Martin mount	Lake Como area (Northern Italy)	Apr. 2010	8
Rialba's tower	Lake Como area (Northern Italy)	Jul. 2010	3
Val Canaria	Canton Ticino (Switzerland)	Oct. 2011	2
Gallivaggio	Valchiavenna (Northern Italy)	Sep. 2012	5
Premana	Valvarrone (Northern Italy)	Oct. 2012	3

This paper significantly extends a preliminary version of the system presented in [3] and is organized as follows. Section II describes the related literature, whereas Section III introduces the system architecture for the proposed monitoring system. The hardware, operating system (OS), communication, service, and intelligent application layers are detailed in Sections IV–VIII, respectively. Section IX describes the experimental results of the proposed system in terms of the application performance and quantifies the gain introduced by the intelligent processing layer.

II. RELATED LITERATURE

Although the collapse of rock faces represents a harmful natural hazard, the lack of clear forerunners and the difficulty of the deployment limit the number of existing available monitoring systems. To the best of our knowledge, there is only one other recent system acquiring acoustic emissions for the monitoring of mountain walls [4]. Therefore, in order to provide a more comprehensive state of the art, we also present monitoring/forecasting systems based on wireless solutions addressing similar scenarios (e.g., structural monitoring and seismic or volcanic eruption monitoring). The comparison focuses on some key aspects, i.e., the nature of the local processing and the distributed processing, the sampling rate, the availability of synchronization mechanisms, and the reprogrammability ability.

An acoustic emission monitoring system designed to operate on Alpine rock walls was presented in [4], where a threshold-based triggering mechanism is used to extract acoustic events. Features are extracted from events and remotely transmitted to a control room only upon request. The sampling rate is 500 kHz. The details about the considered synchronization algorithm are not provided, and reprogramming mechanisms are not considered in this system.

A heterogeneous system for structural health monitoring based on wireless sensor network (WSN) units and wired cameras was suggested in [5]. The software architecture relies on TinyOS [6]; the sampling frequency is 100 Hz. A threshold-based detector is considered at the unit level, and the synchronization relies on the timing-sync protocol for sensor networks [7]. No reprogramming mechanisms are considered.

WISDEN [8] relies on a WSN solution for structural health monitoring, i.e., it is based on TinyOS, and the sampling frequency is 160 Hz. The system relies on a threshold-based event detection mechanism and exploits a wavelet-based compression technique to reduce the communication bandwidth.

WISDEN implements an *ad hoc* data time-stamping scheme for the synchronization of the acquired data. No reprogramming mechanisms for the units are available.

A WSN-based structural health monitoring system named SENTRI was presented in [9]. The considered OS is TinyOS, and the sampling frequency is 200 Hz. Due to the peculiar in-line architecture (the system is organized as a line of 64 sensors), SENTRI relies on a 64-hop routing protocol. The synchronization mechanism is based on a flooding algorithm [10]. Units cannot be remotely reprogrammed.

The wireless intelligent sensor and actuator network (WISAN) [11] is another intelligent WSN for structural health monitoring. Different from previous solutions, WISAN relies on computational intelligent mechanisms and wavelet compression to make network units autonomous and to reduce the bandwidth consumption, respectively. The system relies on an *ad hoc* scheduler, whereas the sampling rate is 50 Hz.

A time-synchronized and reconfigurable WSN for structural health monitoring was described in [12]. There, the network units run FreeRTOS as the OS; the sampling frequency is 200 Hz, and the event detection relies on a modal analysis. Acquisition parameters can be updated during the operational life through remote commands. An *ad hoc* synchronization algorithm, which is called μ -Sync, guarantees very strict synchronization among the network units (below 10 μ s).

Terrascope is a down-hole seismic monitoring system proposed in [13]. It is composed of independent sensing units sampling at 250 Hz and connected to a gateway through a fieldbus. The units rely on a customized scheduler, and the gateway runs a Linux OS. Terrascope allows updating the code running on the units by means of an additional digital bus. Events are detected through a trigger-based algorithm, and clock synchronization relies on a Global Positioning System timer.

A WSN-based monitoring system for volcanic eruptions was described in [14]. The system is based on TinyOS, and the sampling frequency is 100 Hz. The synchronization mechanism is based on a flooding algorithm. The system allows updating its operational parameters through the execution of remote commands, whereas the detection of events is based on the comparison between two exponentially weighted moving averages of incoming signals.

From literature, it comes out that existing solutions for structural health and environmental monitoring are generally not able to satisfy the sampling frequency as required by a rock-collapse forecasting application. In addition, remote reprogrammability and intelligent solutions for balancing energy consumption and the application performance are seldom considered, hence still representing an open and cutting-edge research challenge. In contrast, the rock-collapse forecasting system proposed here is able to acquire and process signals with sampling rates of up to 2 kHz, and it encompasses both basic reprogramming mechanisms and intelligent processing, as detailed in Sections VII-A and VIII, respectively.

III. SYSTEM ARCHITECTURE

The proposed rock-collapse forecasting system is composed of a *remote monitoring system* (RMS) deployed on a rock face and a *control room* that collects data from the RMS for subsequent storage, processing, and interpretation. As shown in Fig. 1, the RMS is composed of a set of sensing and processing units (SPUs) that are connected through a control area network

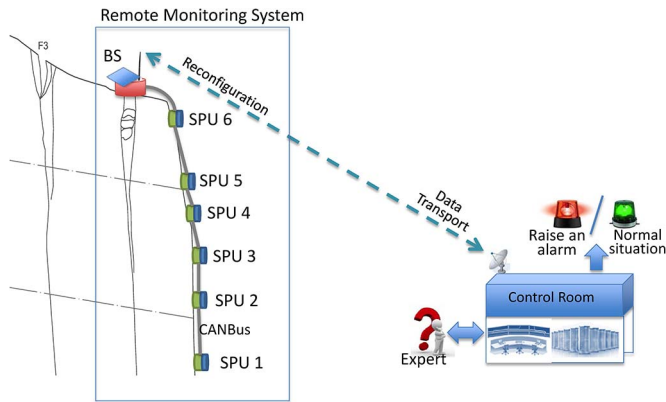


Fig. 1. Proposed rock-collapse forecasting system.



Fig. 2. SPUs in two deployments. (a) Val Canaria. (b) Rialba's tower.

bus (CAN bus) to a BS. SPUs are endowed with sensors to acquire microacoustic emissions (e.g., microelectromechanical systems (MEMS) accelerometers and geophones) and more traditional sensors (e.g., inclinometers, temperature sensors, and strain gauges). The BS collects data from the SPUs and transmits them to the control room.

From the hardware point of view, the proposed rock-collapse forecasting system is characterized by a large heterogeneity in terms of devices, technologies, and performance. SPUs are low-power high-performance digital signal processing units able to acquire, locally process, and extract the information of interest from the data stream sampled at 2 kHz. Examples of SPUs are shown in Fig. 2. The BS provides both the power supply and the synchronization of the SPUs by means of the CAN bus, whereas the control room is composed of PCs and servers organized into a typical internet-based service-oriented network architecture. From the software point of view, the system reflects the hardware heterogeneity where several different software modules and mechanisms cooperate to achieve the targets of the forecasting application.

With reference to Fig. 3, the architecture of the system comprises the following:

- 1) the *hardware layer*, which provides the physical mechanisms for data acquisition, processing, and transmission;
- 2) the *OS layer*, which provides basic software functionalities to higher levels and mainly consists in the OS services;

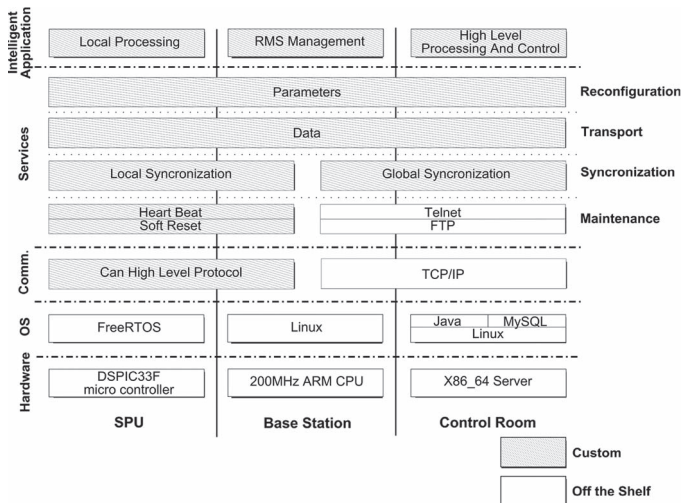


Fig. 3. System architecture of the proposed rock-collapse forecasting system.

- 3) the *communication layer*, which provides the basic mechanisms supporting the local communication between the SPUs and the BS via the CAN bus and the remote communication between the BS and the control room by means of a WiFi/universal mobile telecommunications system (UMTS) protocol;
- 4) the *service layer*, which provides those services supporting the system reconfiguration, the data transport, and the clock synchronization functionalities;
- 5) the *intelligent application layer*, which makes available those functionalities allowing the monitoring application to be adaptive and autonomous.

Due to the peculiar application constraints and the large heterogeneity of the devices, the design of the software required an engineering approach where each software module (together with the interaction with other modules) has been carefully developed and tested from the functional and energetic point of views. To achieve this goal, the designed system relies on a layered architecture, where each layer exploits the functionalities provided by lower layers to expose functions and mechanisms to upper layers. Unlike a vertical “cross-layer” system design, where functionalities are tailored to the specific hardware and layers are cross designed to optimize performance (e.g., aggregation and fusion), a layered architecture is easier to design, test, and maintain. In fact, each layer can be tested separately (black-box testing), and an internal modification (not involving the interfaces) does not affect the other layers. Moreover, a layered architecture allows us to easily reuse the code in applications running on different hardware, whereas “vertical” approaches are generally application specific and hardware dependent.

As a final remark, we emphasize that each layer is designed to offer its functionalities to upper levels, minimizing both computational complexities and memory requirements. The layers of the proposed architecture are detailed hereinafter.

IV. HARDWARE LAYER

As described in the previous section, the proposed rock-collapse forecasting system is based on a set of SPUs that are connected through the CAN bus to the BS, whose goal is to

remotely transmit the acquired information to the control room through a wireless connection.

The SPUs gather sensor measurements, perform local processing to extract features from the acquired data, and apply local classification to only select those microacoustic signals that are worth transmitting to the BS. For this reason, the SPUs must always remain active (i.e., a 100% duty cycle), whereas the BS can be switched off to reduce the energy consumption (the adjustable duty-cycling mechanism of the BS is described in Section VIII-B).

The SPUs are composed of two boards and a set of sensors. The first board, which is based on the 40-MHz Microchip DSPic33F microprocessor (256-kB ROM and 30-kB RAM), performs the processing and communication tasks. The second board contains the signal conditioning circuits for the envisaged sensors. The sensor set comprises MEMS accelerometers and geophones for microacoustic burst inspection, as well as more traditional sensors such as a temperature sensor, an inclinometer, and a strain gauge. The SPUs do not have local energy harvesting capabilities and are powered by the BS through the CAN bus power lines.

The BS plays a fundamental role in the RMS since it coordinates the communication activity among the SPUs and acts as a gateway between the RMS and the control room. The hardware platform of the BS comprises the main board (based on a 200-MHz ARM9-based PC104 board with 32-MB RAM) executing the application at the BS level, the energy harvesting and management boards for energy acquisition through photovoltaic cells and energy management, the shutdown/wake-up board, and the radio module (a radio link/3G UMTS modem). The BS is endowed with rechargeable batteries (12-V 40-Ah lead acid batteries) and solar panels (polycrystalline 20-W nominal panels).

Finally, the control room has a radio module to communicate with the BS and hosts a database and an application server exposing typical service-oriented network facilities.

Further details about the hardware specifications and the energy consumption can be found in [2].

V. OS LAYER

The OS layer optimizes the use of hardware resources and provides functionalities to the upper levels of the system architecture.

Among the wide range of OSs for embedded systems (e.g., see [6], [15], and [16]), we selected FreeRTOS [17] for the SPUs since it is multitasking, real time, modular, and officially supported by the considered microcontroller. These features perfectly fit the needs of our application.

As described in Section IV, the main board of the BS relies on a PC104-based embedded computer; for this reason, we decided to adopt Linux as the OS. Specifically, we configured a custom Debian distribution for ARM (based on kernel 2.6) to get fast boot and shutdown (i.e., 1.9 s) and a small flash memory occupation (~100 MB).

The OS layer of the control room relies on a standard Debian GNU/Linux 5.0 distribution for X86_64 microprocessors.

VI. COMMUNICATION LAYER

The aim of the communication layer is to provide data structures and services that allow the transmission of data and

parameters among the SPUs, the BS, and the control room. The communication layer is organized into two modules addressing the *local communication* between the BS and the SPUs and the *remote communication* between the BS and the control room.

A. Local Communication

Local communication relies on the CAN fieldbus that represents a standard for wired transmissions in several critical applications such as industrial process or automotive control [2], [18]. We selected the CAN bus among different technological solutions, e.g., RS485, Industrial Ethernet, the Process Field Bus (ProfiBus), and ControlNet, for two main reasons. First, it is an open standard. Second, CAN hardware controllers are embedded in many off-the-shelf microcontrollers and industrial PCs (as such, it does not require additional hardware and software modules, e.g., such as in the ProfiBus).

The CAN hardware controllers available on the market generally provide physical and media-access-control layers, whereas there are several options for the CAN-bus-based routing protocol, such as CANopen [19], DeviceNet [20], and CAN Kingdom [21]. Unfortunately, these routing protocols suffer from two main problems. First, they generally rely on a master-slave mechanism, with a special node or a *master* providing all services needed to control the network. Other nodes, or *slaves*, only send data to the master node. Unfortunately, this approach requires the master to be always active, and duty-cycling energy-saving approaches cannot be considered for the master node (as aforementioned in Section III, the BS requires a duty-cycling mechanism to reduce the energy consumption). Second, CAN bus routing protocols are generally designed to deliver a single-packet message accounting for an 8-byte payload for a CAN message. This is a critical point since the amount of data to be transmitted on the CAN bus is up to 8 kB (containing the acquired measurements and the system status information).

To solve the aforementioned problems, we designed an *ad hoc* routing protocol that is able to effectively manage the wired communication between the SPUs and the BS, and, at the same time, keep under control the energy consumption. The proposed protocol is based on a polling approach, with the BS acting as the master node, i.e., each wired transmission is started by the BS, which can either retrieve data from a SPU (the *PULL-data protocol*) or dispatch a parameter update to a SPU (the *PUSH-data protocol*). The BS periodically queries all the SPUs of the network one after the other. The polling phase ends when the BS has completed the PUSH/PULL protocols for all SPUs.

We divided the data into CAN messages by creating the concept of a *transaction* and building a framework similar to the transmission control protocol (TCP).

In the PULL-data protocol, whose unified modeling language (UML) sequence diagram is shown in Fig. 4(a), the BS initiates the transaction by sending the PULL_REQ message to a SPU, which replies with the PULL_REQ_ACK message specifying the number of data packets p to be transmitted. Then, the BS sends the PULL_START message to notify the SPU that it is ready to receive the data frames. For each FRAME_MSG sent by the SPU, the BS acknowledges the receipt with FRAME_ACK. We emphasize that both messages specify the sequence number of the data frame that is currently

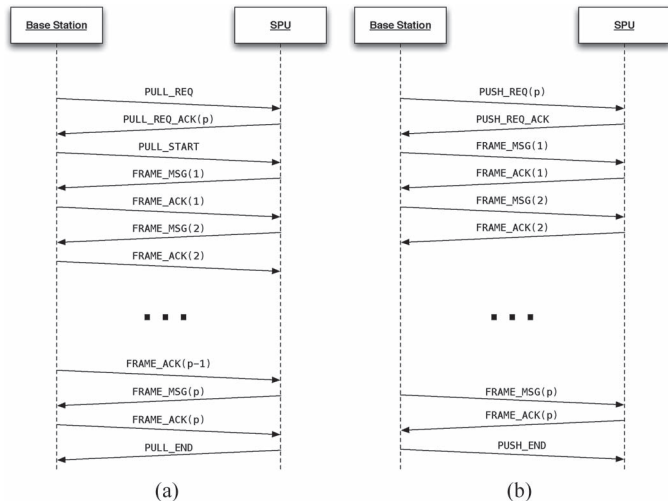


Fig. 4. UML sequence diagrams of the proposed PULL-data and PUSH-parameter routing protocols for the CAN bus. (a) PULL-data routing protocol. (b) PUSH-parameter routing protocol.

transmitted or acknowledged. After the acknowledgment of the last data frame, i.e., the SPU receives the FRAME_ACK associated with the last FRAME_MSG, the SPU sends the PULL_END message to the BS to terminate the transmission. Afterward, the SPU exits the protocol. Similarly, after the receipt of the PULL_END message, the BS exits the protocol. The *PUSH-parameter protocol* is the dual of the PULL-data protocol, and its UML sequence diagram is shown in Fig. 4(b).

B. Remote Communication

The remote communication is based on a long-range wireless transmission making use of either a WiFi-dedicated radio link or a general packet radio service/UMTS mobile-based link. This remote communication module relies on a standard TCP/IP endowed with a *tunneling* procedure aiming at encrypting data and parameters (during the transmission) and creating a virtual private network between the BS and the control room.

VII. SERVICE LAYER

The service layer provides the reconfiguration, data transport, and synchronization functionalities to the sensor network.

A. Reconfiguration

Several parameters are associated to each SPU, allowing the units to undergo reconfiguration both at the data acquisition and signal processing phases. Likewise, the BS can be reconfigured through a set of parameters, which is mainly associated with the duty-cycling activity. The *reconfiguration* service permits the operator to modify the SPUs and the BS parameters, and hence, the interaction of the system with the environment, as well as its internal functional behavior.

This feature is particularly relevant in systems operating in harsh environments and, even more, when little *a priori* knowledge about the phenomenon under investigation is available. For instance, we can modify the parameters associated with the intelligent application layer (as shown in Table III), activate/disable acquisition channels for energy consumption

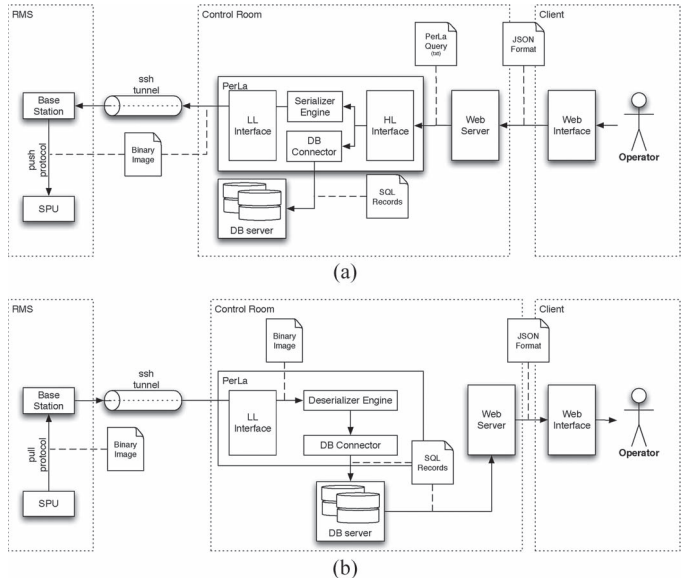


Fig. 5. Reconfiguration and data transport service of the service layer. (a) Reconfiguration service. (b) Data transport service.

reduction, and modify the sampling rate depending on the carried information content. An overview of the reconfiguration service workflow is presented in Fig. 5(a).

The reconfiguration service is composed of the following three modules.

- 1) The operator front end: a web application that, in execution at the control room and as it is accessible to authorized users, permits to change the system parameter configuration through a web browser. By means of a web interface that shows the set of tunable parameters, the operator can select the target unit to be updated and the new set of parameters.
- 2) The middleware: a JAVA application running at the control room that transforms the operator decisions into commands to be sent to the remote system. Among the available middleware for sensor networks [22]–[25], we adopted PerLa [26]. PerLa, which was designed in our institute, is a general-purpose middleware solution for distributed systems aiming at presenting a standard and abstract interface to the data and parameters of network devices. Different from other middleware solutions, PerLa provides a databaselike abstraction for the network elements and guarantees full support for the heterogeneity at the hardware/sensor level, both at the run time and at the deployment time.
- 3) The back end: the software, running at the BS, enables both the update of the parameters affecting the BS and the dispatch of those to be delivered to the target SPU. More specifically, the parameters are only updated if they are different from the previous values. To transmit the new set of parameters, we rely on the PUSH-data protocol described in Section VI.

B. Data Transport

Data transport is the service responsible for delivering the acquired data from the SPUs to the control room. The service workflow, as described in Fig. 5(b), provides mechanisms to address the SPU-to-BS and BS-to-control room transport phases.

In the SPU-to-BS phase, the data transport is based on the PULL-data protocol described in Section VI-A. The data frame represents the image of the in-memory structure used by the SPU to store the acquired measurements.

In the BS-to-control room phase, the BS connects to the control room via the TCP-IP channel described in Section VI-B and transmits the data frames collected from the SPUs.

C. Clock Synchronization

The clock synchronization among the SPUs of the RMS is crucial to guarantee an effective analysis of the gathered data in high-frequency applications. In fact, the localization of a microfracture within a rock face relies on the analysis of highly synchronized microacoustic bursts (i.e., with a maximum clock skew of 1 ms).

The complexity of the hardware architecture of the proposed forecasting system forced us to address the problem of synchronization into two separate phases as follows.

- 1) *Global synchronization*: The SPUs, the BS, and the control room are synchronized by means of the *network time protocol* (NTP) [27] whenever the BS remotely connects to the control room.
- 2) *Local synchronization*: The SPUs are synchronized by means of an *ad hoc* synchronization protocol in between two connections of the BS with the control room. Remarkably, the synchronization among the SPUs can take part, even when the BS is switched off for duty cycling.

The global synchronization mechanism guarantees that all SPUs are synchronized with the clock of the control room whenever the remote communication is established. To accomplish this task, the BS and the control room rely on an NTP client for clock synchronization (i.e., the NTP client connects to the NTP server to get the *Internet clock*). Afterward, the BS propagates the updated time to all the SPUs through a high-priority broadcast message that, once received, permits the SPUs to update their internal clocks.

Differently, the local synchronization guarantees the synchronism among the SPUs. In fact, when the BS is switched off during duty cycling, the SPUs keep on synchronizing themselves to minimize the clock skew by means of the local synchronization mechanism, i.e., one of the SPUs, which is the *synchronization master* that can be either fixed in hardware or remotely defined by the control room, periodically broadcasts a synchronization message to all the SPUs.

In both the local synchronization and the global synchronization, we introduced a mechanism to compensate for the delay caused by the propagation time of the synchronization messages on the CAN bus. Remarkably, this delay T_d is a deterministic offset that depends on the propagation time of the broadcast message on the CAN bus. This delay has been considered common to all SPUs since we experimentally evaluated that the difference of the propagation delays among the SPUs was negligible.

Delay T_d can be easily computed since it depends on the number of bits N_{CAN2b} of the synchronization message, transmission baud rate f_{br} , and the message processing time T_{IRQ} of the SPU microcontroller, i.e.,

$$T_d = \frac{N_{CAN2b}}{f_{br}} + T_{IRQ}. \quad (1)$$

TABLE II
MECHANISMS CONSIDERED IN THE INTELLIGENT APPLICATION LAYER AT THE DIFFERENT SYSTEM UNITS

Unit	Prior Information	Solution
SPUs	Micro-acoustic emissions related to fractures have large magnitude	Bursts are stored depending on a trigger on signal energy applied to running windows
BS	Micro-acoustic emissions related to fractures yield several bursts	Counting bursts arrivals in a short time interval to anticipate the transmissions to the Control Room
Control Room	Burst related to fractures have distinctive shapes	LDA in feature space to discriminate false alarms and possible fractures

T_{IRQ} is generally negligible compared with N_{CAN2b}/f_{br} because the interrupt routine associated with the synchronization messages is executed in about 500 clock cycles (with a 40-MHz clock). The size N_{CAN2b} of the synchronization messages is 130 bits, including the CAN bus identifier, the payload, the cyclic redundancy check, and other fields. As an example, by relying on (1), propagation delays T_d with $f_{br} = 125$ kb/s and $f_{br} = 250$ kb/s are 1.04 and 0.52 ms, respectively. To compensate for this delay, in the interrupt routine, the value of T_d corresponding to the specific baud rate is added to the time stamp received from the master sync (both local or global).

It is noted that, for burst localization purposes, the global synchronization is less important than the local synchronization. In fact, the local synchronization is fundamental to keep the SPUs synchronized to correlate the acquired microacoustic emissions (e.g., by estimating the arrival time of the acquired burst). Clearly, a common drift for all the SPUs does not affect this analysis, which, on the contrary, can be totally compromised by a clock skew among the SPUs. For this reason, we opted for a simple NTP mechanism for the global synchronization.

The timing format is the coordinated universal time (UTC) [28] measured with a granularity of the microsecond. A more effective two-way synchronization procedure, e.g., see [29], could be considered in more critical situations requiring very strict synchronization.

VIII. INTELLIGENT APPLICATION LAYER

The application layer provides intelligent tools for acquiring, transmitting, and processing microacoustic emissions at the SPU, BS, and control room levels. Table II summarizes the intelligent mechanisms adopted by the application layer.

The SPUs implement a trigger mechanism based on sliding windows to only store microacoustic emissions characterized by a suitable magnitude since fractures typically yield an energy content that is far larger than most of the background signals. All the SPUs convey bursts, i.e., microacoustic emissions stored at the SPUs and delivered through the whole system, to the BS, which is in charge of delivering them to the control room over the wireless channel. When the BS is in a *normal transmission mode*, these communications are scheduled in a periodic way to reduce the number of wireless radio activations because of energy constraints. However, the BS might switch to an *alarm transmission mode* when a large number of bursts arrive from the SPUs within a short interval of time, as this might indicate a significant activity within the rock face and possibly a critical situation. In these cases, bursts are delivered to the control room

TABLE III
MAIN PARAMETERS OF THE INTELLIGENT APPLICATION LAYER

Param.	Description	Device	Default value
l	Length of the long window for event detection	SPU	32 samples
s	Length of the short window for event detection	SPU	128 samples
Γ_T	Threshold for event detection	SPU	3
L	Number of samples stored in each bursts	SPU	128 samples
W_B	Length of the time window to count recent burst arrivals	BS	180s
d	Transmission period in normal transmission mode	BS	7200s
Γ_F	Threshold on the samples acquired in W_B to switch to the alarm-transmission mode	BS	3
Γ_B	LDA threshold to distinguish between bursts and false alarms	Control Room	3.8

before the scheduled time. The core of the monitoring activity is executed at the control room, where bursts are processed by extracting features to automatically identify false alarms and submit only bursts that possibly indicate fractures to the visual inspection of a geophysicist. A suitable feature subspace for burst separation is learned by means of linear discriminant analysis (LDA) [30], which is computed over a training set of supervised bursts.

These intelligent mechanisms are described and discussed in the rest of this section, whereas Table III summarizes their main parameters.

A. Intelligent Application Layer: SPU

The application in execution at the SPUs has been organized to address three main tasks, i.e., data *acquisition*, *processing*, and *communication*. During acquisition, data are acquired by three-axial MEMS accelerometers and geophones, which are sampled at 2 kHz. Data are converted by an internal 12-bit analog-to-digital converter (ADC) and stored into a memory buffer.

The processing phase is activated after the storage of each burst, and it consists of digital filtering, followed by a triggering mechanism. First, each of the three channels of the MEMS/geophone recordings is processed through parametric high-pass finite-impulse response (FIR) filters. Then, events are detected by comparing the mean squared amplitude of the signal over two different-sized sliding windows, as in [31]. In particular, if we denote by \bar{x}_l the average squared amplitude over a large window of l samples (e.g., $l = 128$) and by \bar{x}_s the average squared amplitude over a short window of s samples (e.g., $s = 32$), an event is detected when \bar{x}_s/\bar{x}_l exceeds a user-defined threshold Γ_T , which has been experimentally set to 3. Larger values would possibly discard true microacoustic emissions, whereas smaller values would result in the unnecessary storage of false alarms (mainly induced by noise), hence causing to quickly fill the SPU memory.

The number of samples stored in each burst is L , which is defined by the sampling rate and the burst acquisition time. At 2 kHz, the values of L are equal to 128, 256, 512, and 1024 for acquisition times of 64, 128, 256, and 512 ms, respectively. If needed, the acquisition time can be eventually extended up to 4 s through downsampling. The maximum number of events

that can be stored is determined by the RAM capacity of the SPU (7460 bytes), i.e., when the memory is about to saturate, only the condensed information of each burst (such as the time stamp, the signal peak value, \bar{x}_s , and \bar{x}_l) is stored.

In addition to MEMSs and geophones, which acquire high-frequency bursts, the SPUs are equipped with temperature, strain gauge, and three-axial clinometer sensors. These sensors provide low-bandwidth signals, which are acquired at regular time instants and processed by means of a parametric FIR low-pass filter.

The application layer has been designed to guarantee the possibility to change the application at the SPUs at the run time through remote commands sent from the control room and the PUSH-data protocol described in Section VII-A. For instance, digital filters could be remotely enabled/disabled, and application parameters (such as l , s , Γ_T , and L) could be updated.

B. Intelligent Application Layer: BS

The application layer at the BS aims at coordinating all the activities of the RMS, i.e., the BS collects data (including bursts) from the SPUs by means of the PULL-data protocol described in Section VII-B. Then, the BS remotely transmits the data to the control room by means of the remote communication protocol described in Section VI-B. In the normal transmission mode, the BS periodically communicates with the control room to reduce the number of radio module activations (thus lowering the energy consumption). Periodic transmissions (e.g., every $d = 7200$ s) guarantee a good tradeoff between the RMS throughput and the energy consumption but might introduce a delay in critical or dangerous situations (e.g., when several microacoustic emissions are acquired in a very short time). Hence, the BS autonomously counts the number of bursts C_F over a window W_B of the most recent time instants (e.g., $W_B = 180$ s). As soon as this value exceeds a threshold Γ_F , the BS switches into the alarm transmission mode and immediately delivers the bursts gathered from the SPUs to the control room. The BS remains in the alarm transmission mode up to when $C_F \geq \Gamma_F$. When C_F decreases below Γ_F , the BS returns to the normal transmission mode. Γ_F has been experimentally fixed to 3. Smaller values of this threshold would result in an unnecessary activation of the alarm modality, whereas larger values would induce a delay in its activation in emergency situations.

In addition, the application at the BS dispatches commands (received from the control room) to specific SPUs and applies power management policies depending on the energy availability (which is measured by its embedded hardware [2]). Period d characterizing the normal transmission mode and threshold Γ_F can be modified from the control room. The application at the BS also checks the exit status of all the utilities to diagnose remote connection failures and other errors. Errors and warnings are recorded in a log file, which is sent to the control room for diagnosis purposes.

C. Intelligent Application Layer: Control Room

The application at the control room provides the data storage (through an open-source relational structured query language (SQL) database MySQL Server 5.1), presentation (through the Spring framework running on top of a Tomcat application

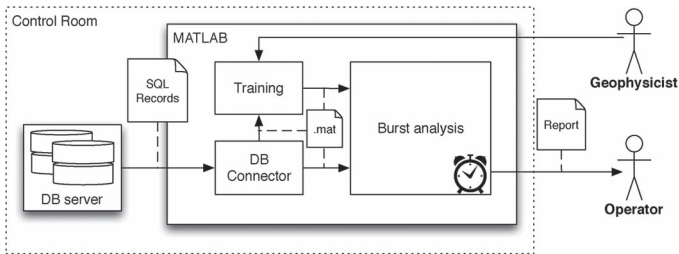


Fig. 6. Burst classification postprocessing workflow at the control room.

server), and intelligent processing of the acquired measurements implemented in MathWorks MATLAB.

The application is composed of a *web application* to remotely visualize the recorded bursts and a local MATLAB application, which implements an intelligent mechanism to separate bursts that can be safely considered *false alarms* from those that are possibly associated to *fractures*, which require the visual inspection by geophysicists. Burst separation is performed by a classifier trained on bursts manually annotated by an experienced geophysicist. The burst classification workflow is depicted in Fig. 6.

The intelligent mechanism at the control room, whose algorithm is shown in Algorithm 1, operates as follows. As soon as a new burst arrives at the control room, it is automatically aligned and then processed to extract relevant features. Then, the LDA [30] is applied to the extracted features to quantitatively assess to which extent the burst indicates a false alarm (that can be safely ignored) or a possible fracture (requiring a geophysicist's intervention).

Input: LDA projection vector a , matrix X_i of a burst to be analyzed (each component is a column)

1. Compute X_i^0 having zero-mean on each component
2. Define the matrix W_i , whose columns are the eigenvectors of $(X_i^0)^T X_i^0$, i.e, the principal components of X_i
3. Compute the aligned burst $S_i = X_i^0 W_i$
4. Compute the feature vector F_i , from the first and third columns of S_i
5. Compute the LDA projection $m_i = a F_i$
6. **if** $m_i < \Gamma_B$ **then**
7. | X_i is considered a false alarm
8. **else**
9. | X_i has to be analyzed by a geophysicist
10. **end**

Algorithm 1: Intelligent Mechanisms to separate bursts.

1) *Burst Alignment:* Let X_i be the 128×3 matrix corresponding to the raw measurements of the i th burst as acquired by the three-channel MEMS of the SPUs (we assume an $L = 128$ measurement per axis, as shown in Table III). Of course, X_i depends on the SPU orientation and location since, in different SPUs, the MEMS axes are set along different orientations. During the alignment, X_i is transformed into a position-independent burst S_i to be compared with the bursts acquired from different SPUs. Such alignment is performed via the principal component analysis (PCA) [32] as follows:

$$S_i = X_i^0 W_i \quad (2)$$

TABLE IV
FEATURES EXTRACTED FROM EACH (ALIGNED) BURST COMPONENT

Time-domain features	Fourier-spectrum features
Mean (before PCA)	Spectrum amplitude
PCA eigenvalue	Main frequency
Signal Amplitude	Variance of spectrum peaks
Signal-to-noise ratio (SNR)	
Peak SNR (PSNR)	
Peak decay	

where X_i^0 is the zero-centered matrix, and W_i is the 3×3 matrix corresponding to the eigenvectors of $(X_i^0)^T X_i^0$, with $(\cdot)^T$ denoting the matrix transpose. Matrix X_i is thus projected into an orthonormal system identified by its principal components, and the aligned burst S_i corresponds to the scores of X_i .

It is worth discussing that the burst alignment by means of a preliminary calibration is not a viable option, i.e., mapping each burst into a common reference axis does not allow a meaningful analysis since the burst orientation depends on the SPU position w.r.t. the fracture source. In contrast, the proposed solution transforms each burst in an adaptively defined orthonormal system, where the burst components are ordered depending on their magnitude (variance).

2) *Feature Extraction:* Features are compact descriptors of bursts, and they provide essential information to determine whether X_i represents a false alarm or a possible fracture. In practice, feature extraction consists in computing a (real-valued) feature vector F_i extracted from each aligned burst S_i .

Nine independent features, which are summarized in Table IV, are extracted from each component of the aligned bursts, i.e., from each column of S_i (exception is made for the mean before the PCA, i.e., the mean of X_i on each axis, which is projected into the principal component space). The features span both the time and Fourier domains. The noise standard deviation, which is used to compute the signal-to-noise ratio (SNR) and the peak SNR (PSNR), is computed as in [33] using the median of the absolute deviation estimator. The peak decay corresponds to the slope of the line joining the highest peak and the second highest peak. In the Fourier domain, the variance of spectrum peaks is introduced as an indirect measure of how far the spectrum is from being unimodal.

We achieved satisfactory discriminative capability by only considering the features from the principal and less significant components of S_i . It follows that each burst X_i is described by an 18-dimensional feature vector F_i .

3) *Burst Separation:* This last phase associates to each burst a measure expressing to which extent it indicates a false alarm or a possible fracture. Such a measure is entirely computed in the feature domain via the LDA configured during the initial training phase.

The LDA is a very popular classification algorithm that performs dimensionality reduction by linearly projecting input samples into the subspace where classes are better separated by a hyperplane. For binary classification problems, the target subspace is 1-D, and the projection corresponds to the inner product against a vector a , which is computed from the training set. Thus, the LDA projects each 18-dimensional feature vector F_i into a scalar measure m_i , which indicates how close F_i is to false alarms or to potentially relevant bursts, i.e.,

$$m_i = a F_i. \quad (3)$$

Then, m_i is used as an indicator of the degree to which the burst

TABLE V
DESCRIPTION, PRIORITY, DURATION, DEADLINE, R VALUE, AND
STACK SIZE OF THE TASKS RUNNING AT THE SPUS

Task ID	Responsibility	Priority	Duration (μs)	Deadline (μs)	R (μs)	Stack size (bytes)
T1	Filtering/event extraction (periodic)	1	267	500	267	420
T2	Parameter update (sporadic)	2	9.6	500	276.6	420
T3	Local synchronization (periodic)	3	110	500	386.6	630
T4	CAN driver (sporadic)	4	81.6	500	468.2	630
T5	CAN bus routing protocol (sporadic)	5	51.4	100000	987.8	1050
T6	Watchdog (periodic)	7	7.8	500000	6499.8	420

is a false alarm, i.e., bursts yielding m_i values below threshold Γ_B can be safely considered false alarms and discarded (as illustrated in Section IX-C). Large values of Γ_B could result in discarding true bursts, whereas low values would result in the processing of a very high number of false-positive detections.

We chose the LDA because its simple model reduces the risk of overfitting the training set; this aspect has to be taken into account as the feature dimension is rather large, and the number of bursts cannot be arbitrarily increased. Of course, the LDA projection vector a can be recomputed whenever additional supervised bursts are available to improve the effectiveness of the burst separation, as in (3).

IX. EVALUATION OF SYSTEM PERFORMANCE

This section aims at evaluating the performance of the proposed system in terms of the schedulability of the tasks in the execution, the clock synchronization, and the intelligent processing.

A. Tasks Running at SPUs: Schedulability Analysis

The *schedulability analysis* verifies whether the execution of all tasks running at the SPUs (as described in Section VIII-A) grants the real-time constraint to be satisfied (or not) [34]. At first, we recall that the considered real-time FreeRTOS is preemptive and based on a rate-monotonic fixed-priority scheduling algorithm. Each task is characterized by a priority value and by execution and deadline times. The priority value is fixed and set at the *design time*. Tasks with a lower value of priority can be preempted by tasks characterized by larger priority values. The execution time, i.e., the time necessary to execute the task, has been computed by considering the assembly language version of the code and the execution time of the involved instructions. The deadline time defines the time instant by which the execution of all tasks must be completed.

The list of the tasks composing the SPU application (ordered by decreasing priority) is given in Table V. Some tasks are periodically executed by the SPU application (e.g., acquisition and processing, local communication, and watchdog), whereas other tasks (e.g., the parameter update and the routing protocol) are triggered by external events such as communication interrupts or timeouts.

In particular, Task T1 represents the acquisition and processing task of the SPU described in Section VIII-A. This task is periodically executed every time a new set of samples is ready to be processed. As such, the deadline time of T1 is determined by the intersampling interval, i.e., 500 μs , which sets a strong constraint; therefore, the task is assigned the highest priority.

TABLE VI
MEMORY OCCUPATION OF THE SPUS

Memory Type	Available (kB)	Occupation (kB)
RAM	30720	30252 (5710 stack segment and 24542 data segment)
ROM	262144	48477 (code segment)

Task T2 is triggered by the PUSH-parameter routing protocol of the reconfiguration service described in Section VII-A. This task updates the parameters of the SPU, and its execution can be postponed after the completion of task T1 but must be completed before a new set of samples is acquired. As a consequence, the deadline time for T2 is set to 500 μs .

Task T3 is only executed if the SPU is a local synchronization master. Similar to T2, this periodic task, which sends the synchronization message to all other SPUs on the CAN bus every 3 s, must terminate before the new set of samples is acquired. Even in this case, the deadline is 500 μs .

The communication task is organized into two subtasks, i.e., T4 and T5. T4 is the CAN driver task, whereas T5 represents the routing protocol for the CAN bus, as explained in Section VI-A. All these tasks are activated with a low frequency. The deadline of T4 is 500 μs to guarantee that each message is sent within the intersampling interval, whereas the deadline for task T5 is 1 s. Finally, task T6 is a periodic task that resets the watchdog of the microcontroller. T6 must end before the watchdog fires, and its deadline is fixed to 0.5 s.

To evaluate the schedulability of the tasks running on the SPU, we rely on the theoretical framework outlined in [34]. For the i th task, we compute the worst case execution time $R_i = C_i + B_i + I_i$, where C_i is the worst case computation time required by the task, B_i is the worst case blocking time, as defined in [35], and I_i is the worst case interference that such task would experience from higher priority tasks. The computation of I_i and R_i is described in [34].

The schedulability of the tasks is guaranteed if all the worst case execution times R_i are below their respective deadline times. As shown in Table V, all the tasks satisfy this constraint, hence guaranteeing the schedulability of the SPU application tasks.

In addition, we evaluated the memory consumption of the SPU application. This analysis is of critical importance since the microcontroller at the SPU is characterized by limited ROM and RAM (as described in Section IV). This led us to consider a manual allocation of data structures in the memory (by modifying the compiler linker script) to exploit the available RAM as much as possible. Table VI summarizes the available and occupied RAM (i.e., the total occupation, the stack, and the data segment) and ROM (i.e., the code segment). We emphasize that the manual allocation of the data structures in the memory allowed us to exploit up to 98% of the available RAM. Moreover, the last column in Table V shows the sizes of the memory stack assigned to each task. It is worth noting that T5 requires the largest amount of stack memory since it implements the PUSH/PULL protocols described in Section VI-A.

B. Synchronization Mechanism

Here, we evaluate the effectiveness of the synchronization mechanism described in Section VII-C. Having this in mind, we designed an experimental setup composed of a BS, three SPUs, and a signal function generator. The BS is connected to the three SPUs through the CAN bus. The three SPUs run

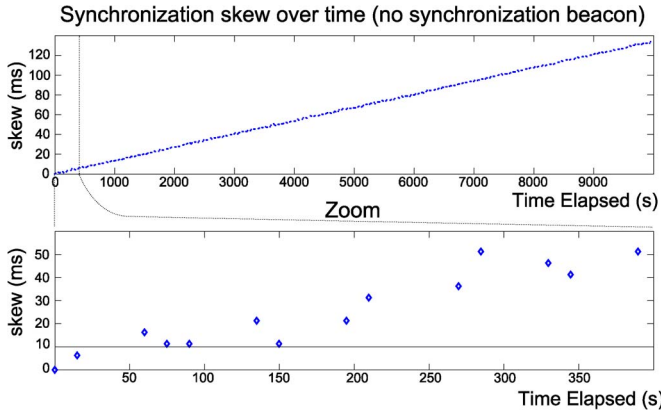


Fig. 7. Clock skew between SPU2 and SPU3 over time. (Top) Time horizon of the analysis is 10 000 s. (Bottom) Analysis focuses on the first 400 s.

the same SPU application with the same parameters. *SPU1* is configured as the local synchronization master, whereas *SPU2* and *SPU3* are configured as slave units. An Agilent 33220A function generator was used to generate a voltage pulse every 15 s with an amplitude of 3.3 V and a time duration of 100 ms. The function generator is connected to both SPU2 and SPU3 through the ADC channel corresponding to a channel of the MEMS accelerometer. With this experimental setup, both SPUs detect synthetically generated events *at the same time*. We evaluated the effectiveness of the proposed synchronization service by measuring the clock skew between SPU2 and SPU3 over time by regularly polling both SPUs and comparing the time stamps associated with the detected events. All the experiments have been performed in our laboratory at a constant temperature.

At first, we considered the case where no synchronization service is employed, and SPU2 and SPU3 have been only synchronized at the beginning of the experiment. These experimental results are presented in Fig. 7. We can observe a clock skew between the considered SPUs, i.e., the internal clock of SPU3 runs slower than that of SPU2, hence increasing the clock skew over time. We recall that, as pointed out in Section VIII, the rock-collapse forecasting application requires the clock skew among SPUs not to exceed 1 ms. Experimental results are interesting since, as depicted in Fig. 7(b) detailing the time horizon of the first 400 s, they show that, without any synchronization mechanism between the SPUs, the constraint on the clock skew (here emphasized by the black line) is no longer satisfied after about 60 s.

Then, we considered a synchronization mechanism with synchronization periods, i.e., the time between the activation of two subsequent synchronization services, ranging from 3 to 100 s and different baud rates for the CAN bus. Fig. 8 shows the experimental results with the baud rate fixed at 125 and 250 kb/s. This figure shows the histograms of the measured clock skew between SPU2 and SPU3 as a result of 300 experiments. Several comments arise. At first, in all the considered configurations of the baud rate, the synchronization period of 3 s guarantees the constraint on the clock skew to be largely satisfied. Second, as expected, the maximum value of the clock skew increases with the synchronization period. These results are in line with those in Fig. 7 and show that the 1-ms constraint is not satisfied with synchronization periods equal or larger than 60 s. For example, given a synchronization period of 100 s, the clock skew between SPU2 and SPU3 can be up to 2 ms.

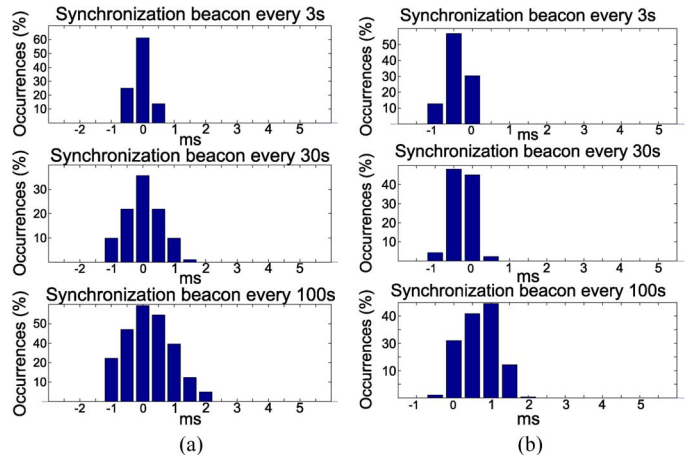


Fig. 8. Histograms for the synchronization skew for different baud rates of the CAN bus. (a) 125 kb/s. (b) 250 kb/s.

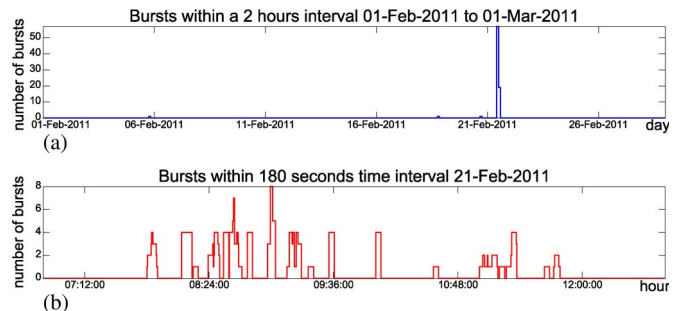


Fig. 9. Bursts recorded by the system deployed at the Towers of Rialba in February 2011. (Top) Number of bursts received by the BS every 2 h (the time between two consecutive connections in the normal transmission mode). (Bottom) Number of bursts received in windows of $W_B = 180$ s.

Third, experimental results show that the baud rate has no (or very little) influence on the measured clock skew. This is in line with what was presented in Section VII-C. In fact, the delay T_d that was induced by the transmission on the CAN bus is compensated via software according to the baud rate of the CAN bus. In our real-world deployments, we opted for a conservative choice of the synchronization period, fixing it to 3 s (the baud rate of the CAN bus is 250 kb/s).

C. Intelligent Processing

We report some analysis on the bursts recorded by the deployment at the Towers of Rialba.¹ Fig. 9(a) illustrates the burst arrivals in February 2011, with each bin reporting the number of bursts recorded in 2 h, which is the temporal distance between two remote communications when the BS is in the normal transmission mode. The highest peak is caused by an artificially induced fracture in the rock face on February 21; Fig. 9(b) details the bursts acquired on this day, reporting the number of bursts on windows lasting $W_B = 180$ s. The setup of the equipment resulted in the generation of two sets of bursts. The first set must be intended as the false alarms introduced by the operators, whereas the last set is associated with the

¹The data set of the acquired bursts is made available to the scientific community at the following web page: <http://troveri.faculty.polimi.it/wp-content/uploads/bursts.zip>.

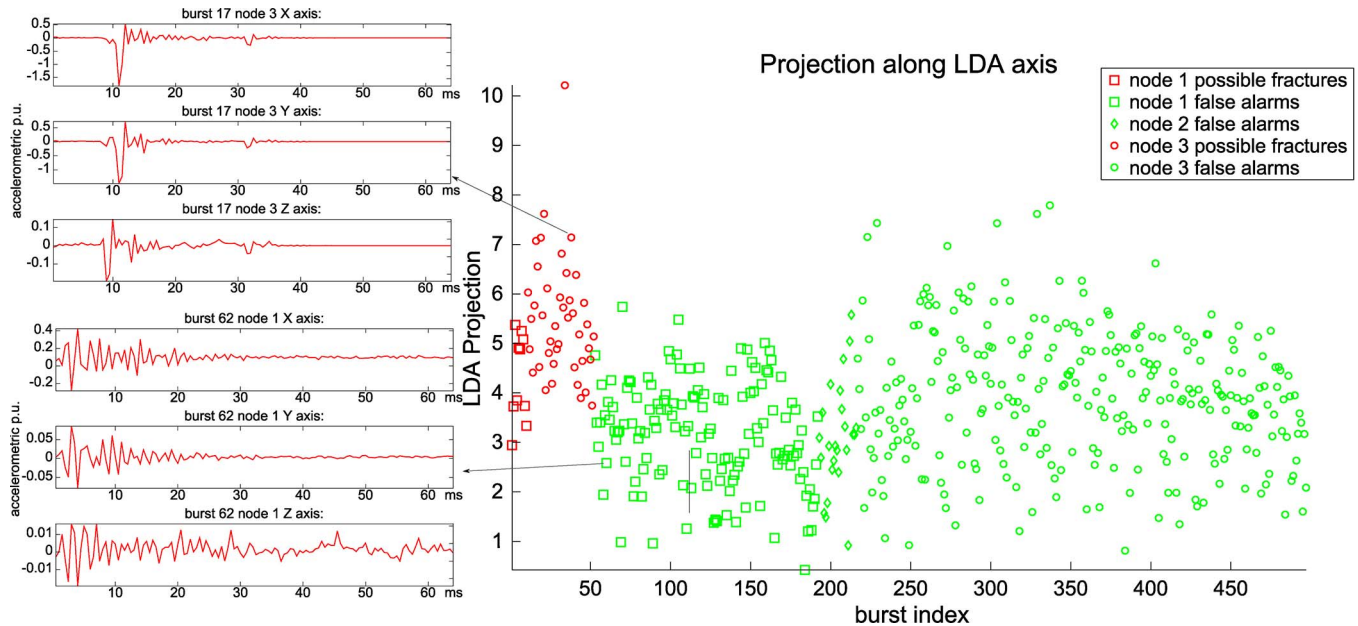


Fig. 10. Burst features projected along the LDA axis. The first 52 markers indicate bursts corresponding to possible fractures, whereas the remaining markers indicate false alarms. Different markers indicate the node that recorded the burst. The LDA values of bursts corresponding to possible fractures are typically larger than those corresponding to false alarms, indicating that these two classes can be separated.

actual fracture; in both situations, the BS entered in the alarm transmission mode.

To illustrate the performance of the burst separation algorithm presented in Section VIII-C, we consider a data set of 497 bursts recorded over more than 2 years of monitoring activity. Geophysicists visually inspected the whole data set, classifying each of them either as bursts indicating false alarms (445 bursts) or possible fractures (52 bursts).

Fig. 10 plots the values of the LDA projection computed on all the bursts of the data set. To ease the visualization, the data set has been organized to report all the possible fractures at the beginning (followed by false alarms) and the group bursts recorded from the same node (i.e., different markers are used for nodes 1, 2, and 3). Therefore, the burst index on the horizontal axis does not reflect the burst time arrival. The vertical axis reports the values of the LDA projection of the respective burst. This figure clearly shows that the separation between possible fractures (corresponding to the initial 52 markers) and false alarms (corresponding to the rest of the markers) in the LDA projections is rather clear. Fig. 10 also shows two bursts (associated with a fracture and with a false alarm, respectively) and their LDA projection values to illustrate the difference between false alarms and possible fractures.

The burst separability is shown in Fig. 11, which presents the empirical distribution of the LDA projection values as computed on the whole data set. This figure suggests that a suitable false-positive filtering strategy is effective and reduces the bursts to be visually inspected by a geophysicist to those yielding an LDA projection above $\Gamma_B = 3.8$. Bursts yielding values of LDA projections smaller than Γ_B can be safely ignored, as these correspond to a false alarm in 98% of cases. Such a selection scheme allows significantly reducing the number of bursts requesting intervention from a geophysicist. The analysis will be hence focused on the remaining 50% of the bursts, taking into account that approximately 19% of these correspond to possible fractures. These percentage can be modified by tuning threshold Γ_B .

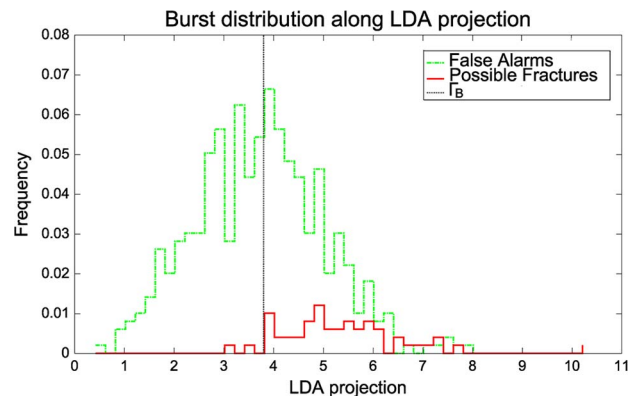


Fig. 11. Distribution of LDA values for bursts corresponding to possible fractures and false alarms.

X. CONCLUSION

The aim of this paper has been to describe a reprogrammable and intelligent rock-collapse forecasting system. The proposed forecasting system guarantees the high-performance processing of high-frequency data directly at the unit level, together with an effective and efficient analysis of acquired signals to identify and extract microacoustic emissions (that represent the possible forerunners of a rock collapse). In addition, the proposed system guarantees both strict synchronization among the acquisition units (which is crucial for the subsequent analyses of data at the control room) and the ability to remotely reconfigure the applications running at the units and at the BS. The system has been successfully deployed in several areas of the Swiss–Italian Alps to forecast rock falls.

ACKNOWLEDGMENT

The authors would like to thank Dr.-Ing. R. Camplani and Dr.-Ing. A. Marullo for their precious contribution in the design and development of the proposed system.

REFERENCES

- [1] O. Bukhalo, "Measuring transformations of acoustic emission signals," *Meas. Tech.*, vol. 45, no. 9, pp. 966–973, Sep. 2002.
- [2] C. Alippi, R. Camplani, C. Galperti, A. Marullo, and M. Roveri, "A high frequency sampling monitoring system for environmental and structural applications," *ACM Trans. Sensor Netw.*, vol. 9, no. 4, p. 41, Jul. 2013.
- [3] C. Alippi, R. Camplani, C. Galperti, A. Marullo, and M. Roveri, "An hybrid wireless-wired monitoring system for real-time rock collapse forecasting," in *Proc. IEEE MASS*, Nov. 2010, pp. 224–231.
- [4] L. Girard *et al.*, "A custom acoustic emission monitoring system for harsh environments: Application to freezing-induced damage in alpine rock walls," *Geosci. Instrum., Methods Data Syst.*, vol. 1, no. 2, pp. 155–167, Nov. 2012.
- [5] A. Basharat, N. Catbas, and M. Shah, "A framework for intelligent sensor network with video camera for structural health monitoring of bridges," in *Proc. Workshops PerCom*, 2005, pp. 385–389.
- [6] P. Levis *et al.*, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*. Hoboken, NJ, USA: Springer-Verlag, 2005, vol. 35.
- [7] S. Ganeriwala, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 138–149.
- [8] N. Xu *et al.*, "A wireless sensor network for structural monitoring," in *Proc. Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 13–24.
- [9] S. Kim *et al.*, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, 2007, pp. 254–263.
- [10] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 39–49.
- [11] E. Sazonov, K. Janoyan, and R. Jha, "Wireless intelligent sensor network for autonomous structural health monitoring," in *Proc. Smart Struct. Mater. Int. Soc. Opt. Photon.*, 2004, pp. 305–314.
- [12] M. Bocca, L. M. Eriksson, A. Mahmood, R. Jäntti, and J. Kullaa, "A synchronized wireless sensor network for experimental modal analysis in structural health monitoring," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 26, no. 7, pp. 483–499, Oct. 2011.
- [13] S. Glaser, "Terra-Scope: A MEMS-based vertical seismic array," in *Proc. SPIE*, 2005, pp. 1–9.
- [14] G. Werner-Allen *et al.*, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Comput.*, vol. 10, no. 2, pp. 18–25, Mar./Apr. 2006.
- [15] M. Farooq and T. Kunz, "Operating systems for wireless sensor networks: A survey," *Sensors*, vol. 11, no. 6, pp. 5900–5930, May 2011.
- [16] C. C. Han, "SOS: A dynamic operating system for sensor networks," in *Proc. Int. Conf. Mobile Syst., Appl., Serv.*, 2005, pp. 163–176.
- [17] FreeRTOS, The FreeRTOS Project, 2011. [Online]. Available: <http://www.freertos.org/>
- [18] W. Lawrenz, *Can System Engineering: From Theory to Practical Applications*, 1st ed. New York, NY, USA: Springer-Verlag, 1997.
- [19] CIA. [Online]. Available: <http://www.can-cia.org>
- [20] DeviceNet. [Online]. Available: <http://ab.rockwellautomation.com/>
- [21] CAN Kingdom. [Online]. Available: <http://www.kvaser.com/>
- [22] K. Aberer, M. Hauswirth, and A. Salehi, "A middleware for fast and flexible sensor network deployment," in *Proc. Int. Conf. VLDB Endowment*, 2006, pp. 1199–1202.
- [23] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, "Tinyrest—A protocol for integrating sensor networks into the internet," in *Proc. REALWSN*, 2005, pp. 101–105.
- [24] J. Sousa and D. Garlan, "Aura: An architectural framework for user mobility in ubiquitous computing environments," in *Software Architecture*, New York, NY, USA: Springer-Verlag, 2002.
- [25] M. Román *et al.*, "Gaia: A middleware platform for active spaces," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 6, no. 4, pp. 65–67, Oct. 2002.
- [26] F. Schreiber, R. Camplani, M. Fortunato, M. Marelli, and G. Rota, "PerLa: A language and middleware architecture for data management and integration in pervasive information systems," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 478–496, Mar./Apr. 2012.
- [27] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.
- [28] "International Telecommunications Union," ITU Radiocommunication Assembly [Online] 2015.
- [29] C. Alippi, *Intelligence for Embedded Systems*. Berlin, Germany: Springer-Verlag, 2014.
- [30] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction: With 200 Full-Color Illustrations*. New York, NY, USA: Springer-Verlag, Feb. 2009.
- [31] C. Alippi, R. Camplani, and C. Galperti, "Lossless compression techniques in wireless sensor networks: Monitoring microacoustic emissions," in *Proc. IEEE ROSE*, 2007, pp. 1–5.
- [32] R. Johnson and D. Wichern, *Applied Multivariate Statistical Analysis*, 6th ed. Upper Saddle River, FL, USA: Prentice-Hall, Apr. 2007.
- [33] D. L. Donoho and J. M. Jonstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, Sep. 1994. [Online]. Available: <http://biomet.oxfordjournals.org/content/81/3/425.abstract>
- [34] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Softw. Eng. J.*, vol. 8, no. 5, pp. 284–292, Sep. 1993.
- [35] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Trans. Comput.*, vol. 39, no. 9, pp. 1175–1185, Sep. 1990.



Cesare Alippi (F'06) received the Dr.Eng. degree (cum laude) in electronic engineering and the Ph.D. degree from Politecnico di Milano, Milan, Italy, in 1990 and 1995, respectively.

He is currently a Full Professor of information processing systems with Dipartimento di Elettronica e Informazione, Politecnico di Milano. He served as a Visiting Researcher with University College London, London, U.K.; the Massachusetts Institute of Technology, Cambridge, MA, USA; the School of Industrial Physics and Chemistry, Paris, France;

and the Chinese Academy of Sciences, Beijing, China. He is the author or coauthor of more than 200 papers in international journals and conference proceedings, and he is the holder of 5 patents. His current research interests include adaptation and learning in nonstationary environments, and intelligent embedded systems.

Dr. Alippi is the Vice President for Education of the IEEE Computational Intelligence Society and an Associate Editor for the *IEEE Computational Intelligence Magazine*. He was an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS and the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENTS from 2003 to 2009, and he served as a member and the Chair of other IEEE committees. He was the recipient of the IEEE Instrumentation and Measurement Society Young Engineer Award in 2004 and the Knight of the Order of Merit of the Italian Republic in 2011.



Giacomo Boracchi received the M.S. degree in mathematics from Università Statale degli Studi di Milano, Milan, Italy, in 2004 and the Ph.D. degree in information technology from Politecnico di Milano, Milan, Italy, in 2008.

In 2004–2005, he was a Researcher with Tampere International Center for Signal Processing, Department of Signal Processing, Tampere University of Technology, Tampere, Finland. Currently, he is an Assistant Professor with Dipartimento di Elettronica e Informazione, Politecnico di Milano. His main

research interests include mathematical and statistical methods for designing intelligent systems, learning in nonstationary environments, and image/video processing.



Manuel Roveri received the Dr.Eng. degree in computer science engineering and the Ph.D. degree in computer engineering from Politecnico di Milano, Milan, Italy, in June 2003 and May 2007, respectively, and the M.S. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in December 2003.

Currently, he is an Assistant Professor with Dipartimento di Elettronica e Informazione, Politecnico di Milano. He served as a Visiting Researcher with Imperial College London, London, U.K. He is the author or coauthor of more than 70 papers in international journals, book chapters, and conference proceedings.

Dr. Roveri is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and served as the Chair and a member in many IEEE subcommittees.