Modeling and Analyzing the Dependability of Short Range Wireless Technologies via Field Failure Data Analysis

Gabriella Carrozza[1,2], Marcello Cinque[1]
(1) Dipartimento di Informatica e Sistemistica, Universita di Napoli Federico II
Via Claudio 21, 80125 - Napoli, Italy
(2) SESM S.c.a.r.l. via C.ne Esterna, Zona ASI, 80014 Giugliano in Campania (NA), Italy
Email: {ga.carrozza, macinque}@unina.it

Abstract— **Direct analysis of failures from the field of application is an effective practice to understand the actual dependability behavior of an operational system. However, despite its wide use over the last four decades on a large variety of systems, field data analysis has rarely been applied to wireless networks. Users accessing the Internet ubiquitously through these networks are increasing, and they expect the same dependability level they already experience on wired networks. But how can we analyze the dependability level of a wireless network? The article defines a novel combined approach to model and analyze the dependability of short range wireless technologies exploiting field data. Through the experience gained from extensive failure analysis of Bluetooth networks, the paper shows how field failure data can play a key role to fill the gap on understanding the dependability behavior of wireless networks.**

Index Terms— **Dependability Modeling, Bluetooth, Wireless Networks**

I. Introduction

Long time has passed since Meyer proposed of the idea of "Ubiquitous Computing", the paradigm which aims at *enhancing computer use by making many computers available throughout the physical environment*, and at making computers *effectively invisible to the user* [1]. Since then, embedded systems engineering and wireless communications have progressed fast, thus making the visionary idea of *Ubiquitous Computing* a reality. The intense device miniaturization and the increasing power of microprocessors, along with the availability of cheap wireless networks and connectivity, allows computers to increasingly pervade everyday human life and activities.
Longer time has even passed since the Internet was anchored to telephone wires and coaxial cables. Since 2005, cell phones have outnumbered PCs and, in the last few years, people access the Internet more from a wireless device than from a wired one, thus enabling mobile Internet access. According to ITU reports [1], mobiles dominate both in quantity and in quality. Small embedded devices have become a daily portable necessity, which is always no more than one meter away from users. PDAs, laptops, cellphones, MP3 players, webcams, and even fridges and microwave ovens, have embedded Internet connectivity, and allow to access the global network from everywhere.
Short Range Wireless (SRW) technologies are at the core of this revolution, as well as the key to ubiquitous networking. They are primarily meant for indoor use and over short ranges, in which they are able to connect portable devices with high connection speed and low power consumption. They are often used at the edges of the wired network, e.g., as wire replacement, to provide mobile users with the last hop to the Internet, from anywhere and at anytime.
Nevertheless, higher mobility means lower speed, as well as worst connection quality in terms of transmission capacity and reliability. Hence, many technical challenges have to be faced in order to serve today customers' demand, who expect the same level of quality they already experience on wired networks. In addition, the wide range of business critical applications in which SRW technologies are protagonists (e.g., mobile banking, mobile commerce, etc), along with their usage in mission critical scenarios (e.g., remote control of robots, rescue of catastrophe survivors, etc.) make it crucial to answer a simple question: *can we rely on these technologies*?
This simple question has not a simple answer. Research efforts in the field of dependability, wireless networks and ubiquitous systems, have to be merged to give a satisfactory response. Indeed, a non-negligible knowledge of the dependability behavior of SRW technologies is required in terms of what are the failure modes, how can we describe/model them, what are the dependability pitfalls and consequences to applications, and how can we face them.
Field Failure Data Analysis (FFDA) is an effective mean to gain the required knowledge. It consists in observing spontaneous occurrences of failures of an operational system, without forcing or inducing artificial failures in the system. The collected failure data provide accurate

This work has been partially supported by the Regione Campania in the context of the projects L.R n. 5/2002, "Reti di sensori senza filo per l'identificazione ed il tracciamento di target mobili" and Misura 3.17 POR 200/2006, "REMOAM - Reti di sensori per il monitoraggio dei rischi ambientali".

[1]International Telecommunication Union, www.itu.int/osg/spu/presentations

information which can characterize the dependability of the system under study.

FFDA has been successfully applied in the last four decades to assess the dependability of operating systems, networked systems, and Internet protocols, as better detailed in section II-C. However, despite the large use in both the academy and the industry, FFDA has rarely been used to characterize the dependability of SRW technologies. In this article we aim to show how field failure data can play a key role to gain the needed knowledge to model the failure behavior and to uncover dependability pitfalls of wireless access networks. The resulting understanding is essential for the effective design of any new solution for dependable wireless networking.

We focus on the Bluetooth technology, which has lots of potential applicability in the "last meter" for personal area networks (PANs) [2]. It has been estimated that in 2005 Bluetooth was a built-in feature for more than 600 million products, manufactured by several companies. CSR (Cambridge Silicon Radio), in its 2007 financial report, said it expects the proportion of new cars that include Bluetooth to increase from 5 up to 30 percent in the medium term. Car-kits use GPS high performance solutions embedded into a Bluetooth chip, thus bringing GPS into a wide range of new low-cost devices.

This article provides an answer to the fundamental question posed above in the context of Bluetooth networks, by exploiting over four years authors' research experience on FFDA of mobile/wireless environments [3], [4], [5]. As explained in Section III, failure data are collected and classified according to the layer they occur, i.e., application, system (Bluetooth stack and operating system), and wireless channel layer, by following both a *user-centric* and a *channel-centric* approach. While the former approach is a well-known practice in FFDA, the latter is based on the novel idea of tracing failure propagation traces from the channel to upper layers, starting from low level causes. To this aim, a novel "merge and coalescence" scheme has been defined. Also, the use of automated recovery actions has been explored to better indicate possible underlying causes of failures.

Conducted experiments allowed to define and to statistically model the failure modes of Bluetooth, classified according to the layer of occurrence, and to characterize failure propagation traces from the channel layer up to the operating system and application layers (see Section IV). In particular, the analysis conducted at the wireless channel layer permitted to define a detailed model of the failing behavior, as described in Section V. It also allowed to study the behavior of Bluetooth channels in the presence and absence of WiFi networks in the area (Section VII). Finally, possible causes of failures are studied (Section VIII), by investigating how failures are typically recovered in our settings.

The results of the experimentation helped to uncover several dependability pitfalls of Bluetooth networks (see Section VI). Some of the key findings are summarized in the following. First, severe failures, such as connection failures and packet losses, may manifest to applications every eight minutes, on average. This is partially due to the bursty nature of observed channel failures, which are more likely to elude integrity checks performed by Bluetooth, hence propagating to the operating system and applications. Second, failures revealed in the absence of WiFi interferences are rarer, but more severe and harder to recover than when WiFi is present. Third, Bluetooth transport layers assume underlying data-link layers to be completely reliable, hence they do not perform error and integrity checks. However, presented results show that these layers are not able to tolerate low level failures.

These findings provide valuable insights that have to be considered when designing Bluetooth-based access networks with demanding dependability and ubiquity requirements.

II. Background and Related Research

A. Bluetooth

Bluetooth (BT) is a short-range wireless technology operating in the 2.4 GHz ISM (Industrial, Scientifical, Medical) band. Two or more (up to eight) Bluetooth enabled devices sharing the same channel form a *piconet*. One of the units acts as the master (the coordinator) of the piconet, the others act as salves. The protocol stack is shown in Figure 1.

The lowest layer is the Bluetooth Radio. It defines

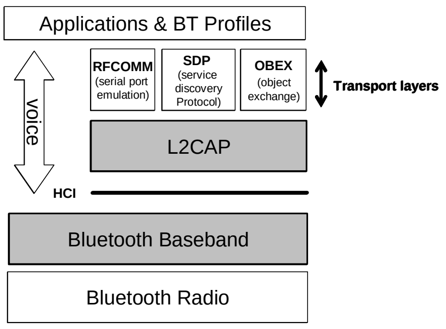

Applications & BT Profiles
RFCOMM (serial port emulation)
SDP (service discovery Protocol)
OBEX (object exchange)
Transport layers
voice
L2CAP
HCI
Bluetooth Baseband
Bluetooth Radio

Figure 1. **Bluetooth Stack**

radio front end, frequency bands, channel arrangements, and receiver sensitivity level. The Baseband layer carries out connections and power control, and it enables two different kinds of physical link, Synchronous Connection-Oriented (SCO), for audio traffic, and Asynchronous Connectionless (ACL), to transport information data. At Baseband layer the channel is divided into time slots (each one of 625 $\mu s.$), and different packet types are defined according to the number of slots they occupy (i.e, 1, 3, or 5 slots).

Baseband includes several *error detectors*, that add redundant information to the packets. The packets are provided with different levels of Forward Error Correction (FEC), Cyclic Redundant Code (CRC), and Header Error Check (HEC). In particular DM*x* (Data Medium Rate, *x* = 1, 3, or 5) packets contain a 16-bit CRC code and use 2/3 FEC to encode information bytes whereas DH*x* (Data

High Rate) packet payload is not FEC encoded. As for the CRC, Baseband adopts the 16 bit CRC-CCITT polynomial $g(x) = x^{16} + x^{12} + x^5 + 1$, which is able to catch all single and double errors, all errors with an odd number of bits, and all burst errors of length 16 or less. It however may fail to detect burst errors which are longer than 16 bits in length, such as 17 bits bursts (with 99.997% coverage) and 18 bits or longer bursts (with 99.998% coverage).

Upon the error detection, Baseband performs the *error recovery* via an ARQ (Automatic Repeat Request) scheme: invalid packets are retransmitted until an acknowledgment is received or a certain timeout expires. By default, the timeout is set to infinite.

Baseband functions are firmware-implemented and accessible via the Host Controller Interface (HCI).

The Logical Link Control and Adaptation Layer Protocol (L2CAP) layer provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. Error correction and flow control are not performed at this layer since the Baseband channel is assumed, by Bluetooth designers, to be reliable.

Several different transport protocols lie over the L2CAP layer, such as RFCOMM, enabling data transfer over virtual serial channels, and SDP (Service Discovery Protocol) providing a mean for applications to discover available services in the piconet.

B. Field Failure Data Analysis

Field Failure Data Analysis (FFDA) of computer systems embraces all fault forecasting techniques which are performed in the operational phase of the life time of a system [6]. This analysis aims at measuring dependability attributes of the actual and deployed system, under real workload conditions.

FFDA studies usually account three consecutive steps: i) data logging and collection, where data are gathered from the operational system ii) data filtering and manipulation, concerning the extraction of the information which is useful for the analysis, and iii) data analysis, i.e., the derivation of the intended results from the manipulated data.

Data logging and collection requires a preliminary study of the system, and its environment, in order to identify the data sources. Typical sources are event logs, i.e., machine-generated log files produced by user applications and system modules. Logs contain information about the regular execution as well as erroneous behavior.

Data filtering and manipulation algorithms are needed to remove invalid data and to coalesce redundant or equivalent data. This is especially true when event logs are used, since they may contain either events not related to failures or multiple events (close in time) which refer to a single failure event; these events need to be coalesced into one failure event.

C. Related work

The importance of FFDA studies of computer systems has been recognized since many years. The first contributions date back to the late 70s, with studies on mainframe systems. In the 90s the research moved its attention to end-user, interactive operating systems and the Internet. As for the former, hangs and the well known "blue screens", found on Windows NT 4 to be mostly due to application failures, were significantly reduced in the successive generation of the OS, Windows 2000, providing the kernel with greater isolation from errant applications [7] [8]. As for the latter, [9] analyzes the causes of failures and the potential effectiveness of various techniques for preventing and mitigating failures in large-scale Internet services.

Currently, we are witnessing an even broader spectrum of research, adding contributions ranging from embedded systems [10] to large-scale and parallel systems [11].

From a detailed study of over than fifty high level technical papers on FFDA, either published by IEEE or ACM journals and conference proceedings, we observed that the most adopted field data sources are event logs (52%), followed by human-generated failure reports (33%), and network monitoring (10%), i.e., the sniffing of the network traffic. Only a small fraction of the related work (5%) uses data coming from more than one source, hence the common practice is to use a single data source.

Despite the mentioned efforts, there is still little experience on the application of FFDA to SRW technologies. The work in [12], proposes a FFDA for a wireless telecommunication system, along with the analysis of failure and recovery rates. However, failure data is relative to the fixed core entities (base stations) of a cellular telephone system. At the same time, we are witnessing an increasing interest of researchers on Bluetooth, especially to its evaluation and modeling. In [13] a collection of user-perceived failure data has been performed. Nevertheless, as also authors stated, the results are not purely scientific in that they have no statistic significance. The work in [14] is concerned with the derivation of an analytical model for the Bluetooth throughput as a function of packet rate. Finally, in [15] a discrete channel simulation of Bluetooth piconets is presented. However, none of these two last works are concerned with dependability issues. In our previous work [3], we proposed the results from a distributed collection infrastructure which enabled a field failure data analysis of Bluetooth systems from the application perspective, whereas in [4] we presented a deeper falure analysis on the Baseband layer. This paper combines the results of previous studies and enrich them with further analysis and insights.

III. A FFDA COMBINED APPROACH

The FFDA of an operational system can be conducted by observing the system according to both a *top-down* and a *bottom-up* approach. The former is a well known practice in the field of dependability evaluation and measurement [9], [6], [3] that allows to infer the

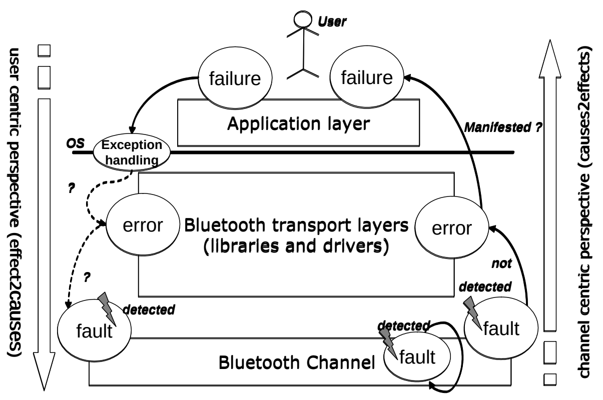


Figure 2. **User- and Channel-centric perspectives**

failure causes starting from the effects on application and Operating System (OS) layers, according to users' point of view (*user-centric* perspective). The latter, instead, is based on the novel idea of tracing how faults propagate to upper layers by directly observing low level causes [4]. With respect to wireless systems, this is a *channel-centric* perspective, in that data communication channel is the starting point for system observation.

Figure 2 emphasizes the differences between the two approaches, with reference to the Bluetooth stack. It is important to precise that, according to the terminology introduced in [16], channel failures can be seen as errors for system failures and as faults for application failures, as well as system failures can be seen as errors for application failures. The user-centric approach allows to analyze failure propagation traces only down to the OS level (Bluetooth drivers failures are logged on system log files). Conversely, by adopting a channel-centric approach, it is possible to monitor failures occurring at the Bluetooth data-link layer, namely Baseband, and to evaluate its *coverage* (i.e., Baseband's ability of self repairing corrupted frames).

In this paper we show how to combine both the perspectives is useful to provide a detailed characterization of SRW technologies dependability. In particular, by following this approach we are able to classify Bluetooth failure modes according to the layer they manifest, and to gain insights into failure propagation traces.

A. Testbed and workload

Field data have been collected by running experiments on a real-world Bluetooth piconet (i.e. a network made up of 1 to 8 Bluetooth nodes, only one of them acting as the *master* or coordinator). Bluetooth piconets can be easily exploited to access the Internet, by means of the Bluetooth Personal Area Network (PAN) profile. A user willing to surf the web with his Bluetooth-enabled mobile phone, starts an inquiry/scan operation to discover other devices in the neighborhood, then - through a SDP operation - he looks for the Network Access Point (NAP). Once the NAP has been found, the user connects to it (note that the connection operation usually takes care of switching the role of the mobile device to slave, letting the NAP be the master of the piconet). Finally, the user can happily navigate to his web-mail inbox.

An application workload (WL) has been designed to emulate the behavior of a typical PAN user. The WL performs all the steps needed to setup the PAN, as mentioned above. The WL then stimulates the wireless channel by transferring data on it. To add uncertainty to piconet evolution, each WL cycle is characterized by several random variables modeling both connection establishment (e.g. whether the inquiry/scan and SDP procedures are performed or not) and channel usage (e.g. according to the random variables which are used to model actual Internet traffic, such as Web surfing, file transfer, e-mail, etc.). Running the WL, and collecting both application and system failures registered on OS log files is useful to achieve the user-centric perspective. During packets transmission, channel level data have been captured by using a Bluetooth air sniffer, in order to achieve the *channel-centric* perspective. The sniffer provided us with all the needed information, from failure reports at the Baseband layer to frame status as they are delivered up to L2CAP and BNEP.

Several experiments have been conducted on the piconet, during a time span of almost two years, collecting more than 140 millions failure data items. In order to investigate the impact of Wi-Fi on Bluetooth failure modes, they have been performed both in presence and in absence of Wi-Fi disturbances.

B. Inferring failure propagation traces

The produced failure data come from multiple sources (WL log files, system log files, and sniffer traces). Data have been properly filtered to discard useless information. In addition, data have to be combined, with temporal coalescence algorithms, to infer failure propagation traces from channel up to system and application layers. To this aim, we propose a manipulation scheme, which we call *merge and coalesce* scheme. The novel aspect of the scheme is to apply the tupling coalescence algorithm, defined in [17], on a merged log file.

The main steps of the scheme are summarized in Figure 3. In the first step, a log file is produced for each node by merging its WL log, system Log, and sniffer traces on a time-based criteria (entries are ordered according to the timestamps written in the logs).

The second step concerns the manipulation of the merged log file by means of the tupling coalescence scheme: if two or more events in the log are clustered in time, i.e, their timestamps are within the same time window, they are grouped into a tuple. The coalescence window size is a crucial parameter that has to be carefully tuned in order to minimize collapses (events related to two different faults are grouped into the same tuple) and truncations (events related to the same fault are grouped into more than one tuple). To this aim, sensitivity analysis can be conducted. The plot in Figure 3, step 2a, shows the typical relationship between the coalescence window

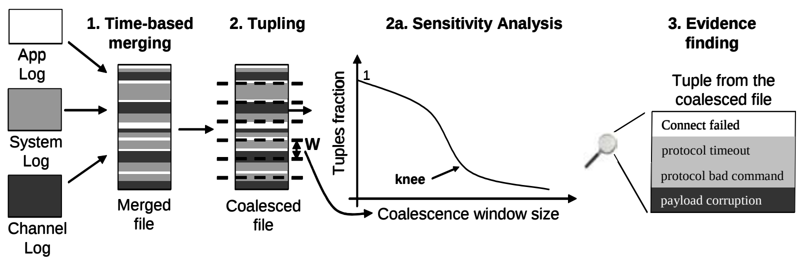


Figure 3. Data coalescence and filtering strategy

size and the tuples fraction. A pronounced knee can be noted on these plots. The knee indicates the sensitivity of the tupling scheme. Choosing a window size before the knee causes truncations: the number of tuples increases, thus events related to the same underlying causes are grouped into different tuples. On the other hand, choosing a value under the knee generates collapses: the number of tuples decreases, thus events related to different causes are grouped into the same tuple.

Failure propagations can then be found by analyzing resulting tuples (step 3 in the Figure). For instance, in the example shown in Figure 3, step 3, the tuple indicates a connection failure due to a bad command issued to the protocol stack, which in turn is caused by the timeout of the protocol due to a packet corruption at the channel level.

C. On the use of recovery mechanisms

The on-field investigation of detection & recovery mechanisms has been extensively used in FFDA to gain insight on the failure causes; examples are [12] and [11]. Recovery mechanisms can indeed reveal if a failure disappears after a partial/total restoration of the state of the system. Hence, they help to pinpoint the corrupted portion of state which potentially led to failure.

The novel aspect introduced in this paper is to explicitly exploit the application workload to embed domain-specific recovery actions. In our case, the following domain-specific recovery actions are triggered subsequently upon failure detection (ordered in terms of recovery cost):

1) *IP socket reset*: the socket is destroyed and rebuilt;
2) *BT connection reset*: the L2CAP and PAN connections are closed and established again;
3) *BT stack reset*: the BT stack variables and data are cleaned up;
4) *Application restart*: the workload is automatically closed and restarted;
5) *Multiple application restart*: up to 3 application restarts are attempted, consecutively;
6) *System reboot*: the entire system is rebooted;
7) *Multiple system reboot*s are attempted.

For instance, BT stack reset and application restart recoveries indicate failures that are due to corrupted values of the state of the Bluetooth stack or to the corrupted execution state of the workload, respectively.

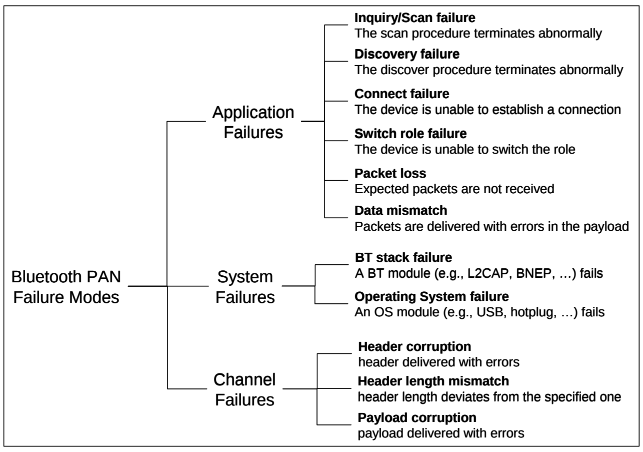

Bluetooth PAN Failure Modes
Application Failures
Inquiry/Scan failure
The scan procedure terminates abnormally
Discovery failure
The discover procedure terminates abnormally
Connect failure
The device is unable to establish a connection
Switch role failure
The device is unable to switch the role
Packet loss
Expected packets are not received
Data mismatch
Packets are delivered with errors in the payload
System Failures
BT stack failure
A BT module (e.g., L2CAP, BNEP, …) fails
Operating System failure
An OS module (e.g., USB, hotplug, …) fails
Channel Failures
Header corruption
header delivered with errors
Header length mismatch
header length deviates from the specified one
Payload corruption
payload delivered with errors

Figure 4. **Bluetooth Failure Modes as they are observed on the field**

IV. Bluetooth failure modes

Field failure data demonstrate to be an effective mean to identify the failure modes of SRW technologies.

In our case, we were able to observe several failure modes and to classify them according to the level in which their occurrence is registered. Observed Failure modes are summarized in Figure 4. Applications exhibit a variety of failures according to the utilization phase where they occur, i.e., inquiry/scan and discovery phases, PAN connection, and data transferring. Failures during the connection can occur either while the connection is set up or while the role of the device is switched from master to slave. Unexpectedly, failures during data transfer, such as packet loss and mismatches in the received data, are experienced, despite error control mechanisms performed by Baseband (see section II-A). As discussed in [18], the weakness of integrity checks is the assumption of having memoryless channels with uncorrelated errors from bit to bit. In the case of Bluetooth, correlated errors (e.g. bursts) can occur due to the nature of the wireless media, affected by multi-path fading and electromagnetic interferences. The failure of the integrity checks is further investigated in the next section.

System level failures are grouped with respect to their location, i.e., Bluetooth software stack and Operating System. Failure types could be further refined according to the component which signals the failure, e.g., L2CAP and BNEP.

Finally, three channel level failures classes have been

TABLE I.
TTF STATISTICAL DISTRIBUTION PARAMETERS

TTF lognormal parameters		
shape	scale	location
1.09	4.42	0.65

identified: (i) Header Corruption (HC) at the Baseband level, (ii) Length Mismatch (LM), i.e., a mismatch in the packet length reported into the Baseband header and the actual one, and (iii) Payload Corruption (PC) at the Baseband level. Channel failures have been also modeled by means of the Markov's chain formalism, as it is detailed in the next section. The model is, in fact, an error-recovery model of the Baseband layer. Actually, it is able to detect and masking HC and LM failures whereas PC failures elude built-in control strategies, and they may propagate to system and application layers.

Failure data also allow to model failure dynamics as stochastic processes. The statistical distribution type then permits to better understand the failure phenomenology. In our case, we attempted to fit the time to failure (TTF) for application failures with three different statistical distributions: the Exponential, the Lognormal, and the Weibull distributions. The fitting has been conducted by means of a statistical software suite, using maximum likelihood estimators and goodness of fit tests. It results that almost all application level failures are distributed as Lognormal. Distribution parameters are summarized in Table I. The Lognormal distribution is used extensively in reliability applications to model failure times. A random variable can be modeled as Lognormal if it can be thought of as the multiplicative product of many small independent factors. In our case, this means that application level failures are the product of many small faults at a lower level. These faults can be both software faults, e.g., heisenbugs (i.e., faults which are activated rarely due to triggers which are not easy to reproduce [19]) at the various level of the Bluetooth stack, and channel faults, as the payload corruption case. Interestingly, only data mismatch failures are distributed as Exponential. This is coherent with the fact that, as will be observed in next section, direct cause for data mismatches are payload corruptions, which also resulted to be exponentially distributed.

The estimation of temporal parameters also allows us to evaluate Bluetooth data communication channel Availability (Av). If we observe Baseband directly from L2CAP, the Bluetooth data communication channel is *available* when it is able to deliver correct frames to the L2CAP layer. During retransmission attempts L2CAP perceives the channel as *not available*. We calculated the channel availability as

$$Av = \frac{MTTE}{MTTE + MTTR} = \frac{139.06}{139.06 + 7.51} = 0.948762 \quad (1)$$

More detailed analysis allows to derive interesting characteristics of the failure behavior. For instance, we attempted to characterize BT connections survivability, i.e., the duration of BT connections before they are unexpectedly lost due to failure (and not due to normal connection closing operation). We observed that the connection duration with respect to failures is statistically *self-similar*, i.e., it shows the same statistical properties at many different scales. On a side, this implies that connection duration times can be modeled with heavy tailored distributions (e.g., the Pareto distribution). On the other side, this shows evidence that connection durations exhibit *long range dependence*: the failure of a connection at a given time is typically correlated with connection failures at all future instants.

V. MODELING CHANNEL LEVEL FAILURES

When dealing with SRW, most of the failure causes lie at the channel layer due to interferences and multipath fading phenomena which are very likely to occur at this level. We tried to model the error-recovery strategy adopted by the Bluetooth tecnology, and in particular by the Baseband level, in order to have a picture of its low-level behavior and to provide useful hints for its reliability improvement. The model is described according to the Markov's chain formalism, and it is shown in Figure 5. Six are the proposed states:

- ***Transmission state*** ***(Tx)***: identifies the proper working mode. When the channel is in Tx state, frames are being delived correctly;
- ***Length mismatch*** ***(L)***: the receiver endpoint received a packet whose length differs from the one reported in the header. The corruption is properly detected;
- ***Header corruption*** ***(H)***: the Baseband packet header is corrupted. The corruption is properly detected;
- ***Payload corruption*** ***(PC)***: the Baseband packet payload is corrupted. The corruption is properly detected;
- ***Fail:*** the Baseband packet payload is corrupted. However, the corruption is not detected and a transmission failure occur;
- ***Retransmission state*** ***(RTx)***: identifies the recovery working mode. Once an error has been detected the erroneous frame is retransmitted.

The channel remains in Tx state until there is no error in frame transmission ("P_{SUCC}"). When a corruption occurs, two transitions are possible: (i) the channel goes into H, L, or PC (with transition probabilities equal to P_L, P_H, P_{PC}) if the corruption is detected, and (ii) the channel goes into $FAIL$ (with a probability equal to P_{FAIL}) if the payload corruption is undetected.

In the first case, Baseband is able to perform a recovery action via ARQ retransmission scheme, hence the channel state moves to RTx and then to Tx. Note that the RTx state is formally equivalent to Tx. However, it has been introduced to improve the readability of the model. In the second case, Baseband does not detect the corruption, thus no recovery action is performed. The frame is delivered with errors to the upper layer (L2CAP), resulting into a failure. After the erroneous transmission, the channel returns to work properly again, i.e., it comes back to Tx.

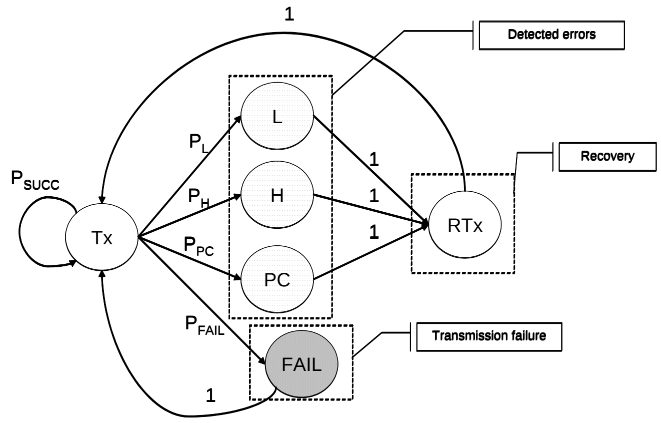


Figure 5. **Markov chain of Bluetooth channel error/recovery model**

The described model is *parametric*: it can be tailored for different working settings by using actual values. These values can be experimentally calculated from the collected data. Being S_i a generic state of the Markov chain, they are the ratio between the number of observed transitions from the state S_i to another state S_j and the number of total transitions out to S_i.

The defined transition probabilities permit also to formalize the Baseband coverage, Cov_B (see section II-A) as in equation 2.

$$Cov_B = 1 - P(Failure|fault) = 1 - \frac{P(Failure \cap fault)}{P(fault)} \tag{2}$$

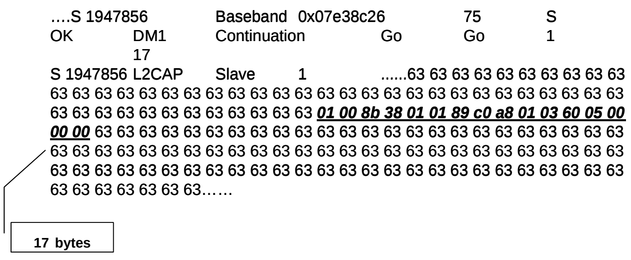
where the fault event occurs when a fault is activated on the channel, hence it can be thought as the union of LM, HC, PC, and undetected PC (i.e., Failure) events. Being the failure event included into the fault event, it results $P(Failure \cap fault) = P(Failure)$, thus it results:

$$Cov_B = 1 - \frac{P(Failure)}{P(fault)} = 1 - \frac{P_{FAIL}}{1 - P_{SUCC}} \tag{3}$$

VI. Failure propagation analysis

As stated in section IV, there exists a class of channel level failures, namely PC, that is able to elude Baseband error control mechanisms, and to propagate to upper layers with a non zero probability. Thanks to field experiments, and to a thorough inspection of packets content, we were able both to observe the occurrence of PC on monitored Bluetooth channel, and even to pinpoint the flipped bits.

A snapshot of a corrupted payload is shown in Figure 6. Note that we were able to uncover this corruption since we forced the WL to transfer a known character sequence with a fixed length, e.g. “CCCC”. The highlighted burst is 136 bits long. This is the reason why it is able to elude Baseband error control mechanisms (see Section II-A). We experienced that the burst length is a random variable, L, with an expected value equals to 512 bits and a standard deviation equals to 646 bits, hence they are longer on average than 18 bits.



```
....S 1947856       Baseband  0x07e38c26          75         S
OK          DM1     Continuation          Go         Go         1
            17
S 1947856 L2CAP     Slave      1            ......63 63 63 63 63 63 63 63 63 63
63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63
63 63 63 63 63 63 63 63 63 63 63 63 01 00 8b 38 01 01 89 c0 a8 01 03 60 05 00
00 00 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63
63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63
63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63
63 63 63 63 63 63 63......
```

Figure 6. **Example of corrupted payload**

Graph in Figure 7 shows how a PC can propagate. On the leftmost side of the graph, it is shown that 99.59 % of PC are detected by Baseband, hence they do not reach upper layers. With respect to the undetected failures, values on the graph links represent the conditional probability of failures given that a PC occurred and eluded Baseband control. Several consequences can then occur, according to the probabilities reported on the graph. In fact, PC can either remain latent (i.e. isolated) at the system level, or propagate to the user level in the form of *application failures*. In the former case, they are *confined* at system level even if no further error controls are performed (in fact, both L2CAP and BNEP assume underlying levels to be completely reliable). The actual induced failure depends on the location of the burst within the transmitted packet, as depicted in Figure 8. As for example, if the corruption affects the L2CAP header, the packet can not be properly decoded. As a consequence, it will not be delivered to upper layers, thus causing a packet loss, i.e. an omission failure, at the user level. Conversely, if the burst is located in the L2CAP payload, the erroneous content can be directly delivered to the application, which may then exhibit a value failure, i.e., a data mismatch in the Figure.

VII. Wi-Fi impact on BT dependability

Many efforts have been devoted to investigate coexistence issues between Wi-Fi and Bluetooth [20]. We tried to estimate how the presence of a Wi-Fi network in the neighborhood can impact Buetooth failure modes. To this aim we let WL run both in the presence and in absence of Wi-Fi interferences.

We compared the conducted experiments in terms of Baseband failure rate and failure distribution over channels.

In the presence of Wi-Fi interferences, the Baseband failure rate has been measured as 6.822 faults per second. Since the average number of transmitted frames per second is 596, this results into a frame error rate of about 0.012 (i.e., about 1 frame out of 100). However, the most of these errors are promptly detected and masked by Baseband’s correction mechanisms, in that its coverage, with respect to all channel failures, has been measured as 0.9996. Undetected failures can be modeled as an exponential random variable with a 458716 ms mean. This means that about every eight minutes a Baseband

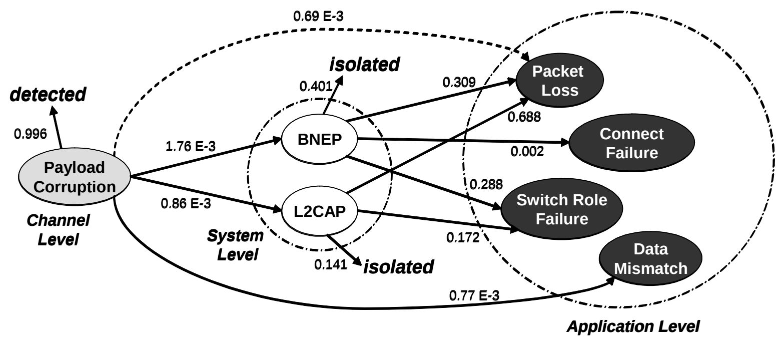


Figure 7. **Propagation phenomenology**

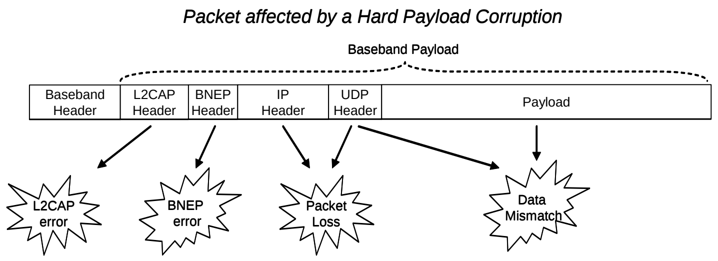


Figure 8. Experienced failures according to the burst location within the packet

error is not detected, and a wrong frame propagates to upper layers. As one could expect, a lower failure rate (equals to 0.516 faults per second) has been experienced when Wi-Fi Access Points (APs) in the neighborhood are turned off.

As for failure distribution across wireless channels, results are shown in Figure 9. In particular, Figure 9(a) shows failure probability for all failures (even detected ones) over channels when WiFi is present whereas results in Figure 9(b) refer to the experiment conducted without WiFi disturbances. In the first case, error probability is highly concentrated over the channels evidenced by dotted lines corresponding to the actual channel overlap between the three Wi-Fi APs deployed in our laboratory and the Bluetooth channels (Bluetooth uses 79 wireless channels, each 1-MHz-wide, in the unlicensed 2.4 GHz band; Wi-Fi uses eleven 22-MHz-wide sub-channels across the same band of Bluetooth; when a Bluetooth and a WiFi radio are in the same area, a single Wi-Fi channel overlaps with 22 of the 79 Bluetooth channels.). Fault probabilities strongly depend on APs usage. For instance, the AP working on channels from 1 to 23 is rarely used, thus justifying the low fault probability over these channels. Figure 9(b) shows that the probability over interfering channels drastically decreases when WiFi is absent. This is a further confirmation of the lower fault rate we measured in the absence of interferences. Interestingly, we found that faults that occurred in absence of Wi-Fi interferences were more "severe" than those that occurred when Wi-Fi is present. This conclusion can be drawn by investigating time to failure statistics for all failures (detected and undetected). In both cases, they fit a Lognormal distribution, but with different values of distribution parameters (see Table II).

TABLE II.
TTF DISTRIBUTION PARAMETERS WRT TO WIFI PRESENCE IN THE NEIGHBORHOOD

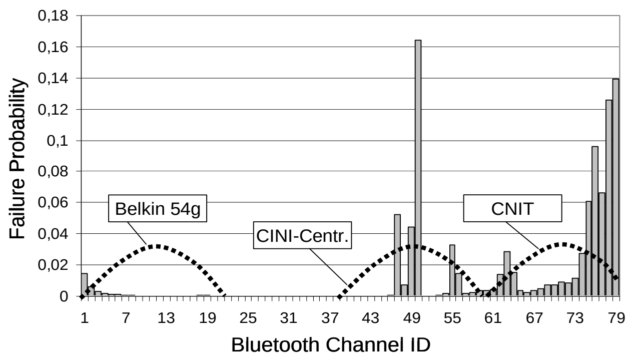
TTF Lognormal Distribution Parameters			
Exp.	shape	scale	location
$WiFiPresent$	1.09	4.42	0.65
$WiFiAbsent$	2.05	5.049	0.937

In the presence of Wi-Fi, faults are mainly due to interferences which tend to be polarized on the overlapped channels. After the occurrence of a failure due to collision, the frame is retransmitted over a different channel. However, the channel might either be free or still occupied by the Wi-Fi interference. This variability causes both short- and medium-length inter-failure times. When Wi-Fi is not present, there are no polarized interferences, or, in other terms, the fault phenomena is spread (e.g., lost of synchronization among nodes or wide-band disturbances). Hence, it is more likely that a retransmission will fail.

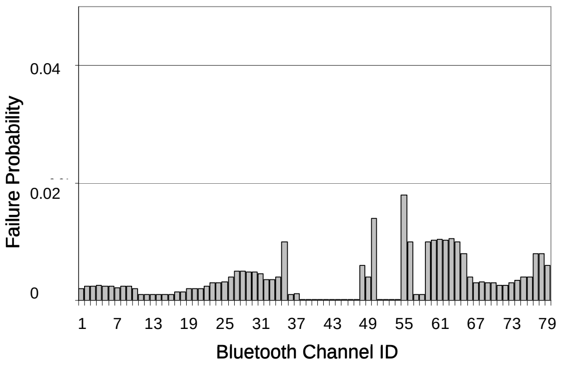
This leads us to observe that in absence of Wi-Fi short inter-arrival times of failures are more probable. In other terms, the absence of disturbances causes the faults to be more clustered in time. The reason for this is to be found into the frequency hopping scheme adopted by Bluetooth.In order to corroborate this intuition, we also investigate Mean Time To Recover (MTTR) in both circumstances. Consistently with above results, MTTR increases when Wi-Fi is not present (from $7.51ms$ to $9.52ms$), i.e. more retransmissions are needed when the fault phenomenon is not polarized. Finally, the Baseband level exhibited a lower capability of detecting failures due to spread phenomena in that its coverage decreases by one

TABLE III.
RESULTS WITH RESPECT TO THE NUMBER OF BT SOURCES

Piconets	Nodes	Error rate	MTTF(ms)	MTTR(ms)	Av	% var.
1	*4*	19.60 (s^{-1})	51.0	7.84	0.889	-6.20
2	*2*2*	9.34 (s^{-1})	107.0	7.93	0.926	-2.40



(a) WiFi present (overlapping zones for each AP are shown)



(b) WiFi not present

Figure 9. **Histogram of failure probability across Bluetooth channels.**

order of magnitude (it passes from 0.9996 to 0.9968). This means that failures due to spread phenomena are more prone to elude Baseband's CRC integrity check.

We also conducted experiments involving more than one piconet, in order to investigate whether the impact of WiFi depends on the number of Bluetooth sources in the neighborhood. In particular,we set up a piconet composed of four Bluetooth nodes, and two different Bluetooth piconets, each composed of two nodes. Table III reports the achieved results. and the percent variation of Av with the reference experiment (i.e., the one with only two nodes in the piconet). As one could expect, the presence of other Bluetooth nodes causes the fault rate to increase. In particular, the higher fault rate is due to the fact that more nodes report the same underlying problem when four nodes are in the piconet. Conversely, in the case of two interfering picontes, the fault rate increase is due to the presence of two master nodes which might choice the same channel over time, causing interferences.

TABLE IV.
APPLICATION FAILURES AND CORRESPONDING RECOVERY ACTIONS

	Recovered by						
Application Level Failure	IP socket reset	BT conne-ction reset	BT stack reset	Application restart	System reboot	Multiple app restart	Multiple sys reboot
Inquiry/scan failed			34.5	30		35.5	
SDP search failed		40.1	39.8		20.1		
Connect failed		0.5	14.9	55.8	25.6	0.1	3.1
PAN connect failed		46.4	35.7	5.4	12.5		
Sw role request failed		28.4	48.2	4.9	17.3	1.2	
Packet loss	5.9	7.2	25.8	33.1	26.7	0.2	1.1
Data mismatch							

VIII. ANALYSIS OF RECOVERY ACTIONS

Table IV reports the relationship between application level failures and recovery actions. Each number in a cell represents the percentage of success of the recovery action (in a column) with respect to a given application level failure (in a row). The numbers give an indication of the effectiveness of recovery actions.
Several understandings can be obtained from the results. As an example, packet losses recovered by an IP socket reset (5.9% of packet losses) are due to "Hard Payload Corruptions" detected by the IP CRC. It is indeed not necessary to reestablish the L2CAP and BNEP connections. The rest of the packet losses are instead likely due to a broken link, since they at least require the connection to be reestabilished. These broken link failures can be caused by "Hard Payload Corruptions" affecting the L2CAP or BNEP headers, then causing the corruption of the data structures that maintain the link state. Hence, depending on the severity of the corruption, several recovery actions are needed, from the BT Connection reset to the reboot of the machine. For "Data Mismatch" failures, no recoveries can be defined, since a real application only relies on integrity mechanisms furnished by the communication protocols.

IX. CONCLUSIONS

Short range wireless technologies are the key of ubiquitous networking. They represent the principal medium to access the Internet from mobile devices. As these technologies are widely used in business and mission critical applications, characterizing their dependability represents a significant issue. Field Failure Data Analysis shows to be an effective instrument to build the needed knowledge on the dependability behavior of actual wireless networks. The case of Bluetooth, analyzed in the article, gives evidence of how field data are useful to model the dependability behavior and to uncover possible pitfalls. In the paper, a novel approach is defined to conduct the FFDA of wireless technologies, which combines the advantages of both the top-down and bottom-up perspective to analyze field data. Exploiting the approach, the dependability of Bluetooth is characterized in terms of i) its failure modes and statistical properties, ii) a markov model for channel level failures, iii) failure propagation traces to system and application level failures.

Presented results are useful to define mitigation means to improve the overall dependability level of Bluetooth networks. The same analysis needs to be conducted on other wireless networks enabling ubiquity both over long distances, e.g., WiMAX, (Worldwide Interoperability for Microwave Access), and within short ranges, e.g., UWB (Ultra Wide Band), and WUSB (Wireless USB), with the aim of building large and publicly available field failure data repositories. These can be exploited by researchers and practitioners to design dependable wireless solutions.

REFERENCES

[1] M. Weiser, "Ubiquitous computing," *IEEE Computer*, vol. 26, no. 10, October 1993.

[2] P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla, "Personal Area Networks: Bluetooth or IEEE 802.11?" *International Journal of Wireless Information Networks Special Issue on Mobile Ad Hoc Networks*, April 2002.

[3] M. Cinque, D. Cotroneo, and S. Russo, "Collecting and Analyzing Failure Data of Bluetooth Personal Area Networks," *Proc. of the 36th IEEE International Conference on Dependable Systems and Networks (DSN'06)*, June 2006.

[4] G. Carrozza, M. Cinque, D. Cotroneo, and S. Russo, "Dependability evaluation and modeling of the bluetooth data communication channel," in *PDP '08: Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*, 2008, pp. 245–252.

[5] M. Cinque, D. Cotroneo, Z. Kalbarczyk, and R. K. Iyer, "How do mobile phones fail? a failure data analysis of symbian os smart phones." in *Proc. of the 37th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2007), Edinburgh 2007*, 2007, pp. 585–594.

[6] R. K. Iyer, Z. Kalbarczyk, and M. Kalyanakrishnam, "Measurement-Based Analysis of Networked System Availability," *Performance Evaluation Origins and Directions, Ed. G. Haring, Ch. Lindemann, M. Reiser, Lecture Notes in Computer Science 1769, Springer Verlag*, 2000.

[7] B. Murphy and B. Levidow, "Windows 2000 Dependability," Microsoft Research, Microsoft Corporation, Redmond, WA, MSR-TR-2000-56, June 2000.

[8] C. Simache, M. Kaaniche, and A. Saidane, "Event Log based Dependability Analysis of Windows NT and 2K Systems," *Proceedings of the 8th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'02)*, December 2002.

[9] D. Oppenheimer, A. Ganapathi, and D. Patterson, "Why do Internet services fail, and what can be done about it?" *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems (USITS '03)*, March 2003.

[10] J. Carlson and R. Murphy, "Reliability Analysis of Mobile Robots," *Proc. of the 2003 IEEE International Conference on Robotics and Automation (ICRA'03)*, September 2003.

[11] B. Schroeder and G. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," *Proc. of the IEEE International Conference on Dependable Systems and Networks (DSN 2006)*, June 2006.

[12] S. M. Matz, L. G. Votta, and M. Malkawi, "Analysis of Failure Recovery Rates in a Wireless Telecommunication System," *Proc. of the 2002 International Conference on Dependable Systems and Networks (DSN'02)*, June 2002.

[13] M. E. D. D. Deavours and J. E. Dawkins, "User-perceived interoperability of bluetooth devices," The University of Kansas, 2335 Irving Hill Road, Lawrence, KS 66045-7612, Tech. Rep., June 2004.

[14] A. Pi Huang and P.Chatzimisios, "Modelling the bluetooth logical link control and adaptation layer," Multimedia Communications Research Group, Bournemouth University, Tech. Rep. MSU-CSE-99-39, November 2004.

[15] A. A. B.B. Rodríguez, M.P. Sánchez Fernández, "Discrete channel smulation of bluetooth piconets," *Workshop on Broadband Wireless Ad-Hoc Networks and Services*, September, 12-13 2002.

[16] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[17] M. F. Buckley and D. P. Siewiorek, "A comparative analysis of event tupling schemes," *proc. of The 26th IEEE International Conference on Fault-Tolerant Computer Systems (FTCS '96)*, June 1996.

[18] P. Koopman and T. Chakravarty, "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks." *Proc. of the 34th IEEE International Conference on Dependable Systems and Networks (DSN '04)*, June 2004.

[19] K. Vaidyanathan and K. S. Trivedi, "A Comprehensive Model of Software Rejuvenation," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 124–137, April-June 2005.

[20] J. Lansford, A. Stephens, and R. Nevo, "Wi-Fi (802.11b) and Bluetooth: Enabling coexistence," *IEEE Network*, pp. 20 – 27, September/October 2001.

Gabriella Carrozza is currently R&D project manager at SESM (Italy) in the field of dependable systems and open source software in the embedded system domain. She graduated in Computer Engineering from University of Naples, Italy, in 2005, where she became PhD in 2008. She is mainly interested into the dependability of off-the-shelf based safety critical systems, especially in the field of Air Traffic Control, as well as on software faults diagnosis.

Marcello Cinque is currently an assistant professor of computer engineering at the University of Naples Federico II. He graduated from University of Naples, Italy, in 2003, where he received the PhD degree in computer science engineering in 2006. His main research interests include on-field dependability analysis of mobile and sensor systems, and middleware solutions for mobile ubiquitous and multimedia systems.