# Collecting and Analyzing Failure Data of Bluetooth Personal Area Networks

Marcello Cinque[1,2], Domenico Cotroneo[1] and Stefano Russo[1,2]
*(1) Dipartimento di Informatica e Sistemistica,Università di Napoli Federico II*
*Via Claudio 21, 80125 Napoli, Italy*
*(2) Laboratorio ITEM- Consorzio Interuniversitario Nazionale per l'Informatica*
*Via Diocleziano 328, 80124 Naples, Italy*
{*macinque, cotroneo, sterusso*}*@unina.it*

## Abstract

*This work presents a failure data analysis campaign on Bluetooth Personal Area Networks (PANs) conducted on two kind of heterogeneous testbeds (working for more than one year). The obtained results reveal how failures distribution are characterized and suggest how to improve the dependability of Bluetooth PANs. Specifically, we define the failure model and we then identify the most effective recovery actions and masking strategies that can be adopted for each failure. We then integrate the discovered recovery actions and masking strategies in our testbeds, improving the availability and the reliability of 3.64% (up to 36.6%) and 202% (referred to the Mean Time To Failure), respectively.*

## 1 Introduction

The wide spread use of mobile computing platforms is leading to a growing interest of dependability issues. Examples are health care [2] and aircraft maintenance systems [16]. Even for mass-market software, such as news and e-commerce services, where the consequences of individual failures are usually not catastrophic, unreliability can have serious commercial implication for vendors and users. Although several research studies have been conducted on the dependability of mobile environments, as discussed in section 2, none of them attempted to identify system bottlenecks and to quantify dependability measures by providing information from field data. These studies can help i) developers to realize more reliable services, and ii) researchers to gain insight into the dependability behavior of mobile environments, and to design abstract models that can be used for further analysis. The Bluetooth wireless technology is nowadays widespread adopted in a variety of portable devices. In particular, the Bluetooth Personal Area Network (PAN) feature offers IP support over Bluetooth, thus expanding the model of personal communications to provide connectivity and Internet access to and between heterogeneous devices. As demonstrated in recent works, the utilization of Bluetooth as a "last meter" access network, represents an opportunistic and cost-effective way to improve the connection availability of the wide-spread existing 802.11 networks [13, 15].

In this work a comprehensive failure data analysis campaign on Bluetooth Personal Area Networks has been conducted. Results are then used to characterize failures distribution and to improve the dependability of Bluetooth PANs. In particular, the novel contribution of this work is twofold.

First, we leverage the dependability level of Bluetooth PANs. To this purpose, we define a failure model of such networks, identifying failures and their statistical distributions. For each observed failure we try to discover its sources, to identify the most effective recovery action and, in some cases, we are able to apply error masking strategies by completely eliminating some failures from occurring. We then integrate the discovered recovery actions and masking strategies in our testbeds, demonstrating that the availability and the reliability can be improved of 3.64% (up to 36.6%) and 202% (referred to the Mean Time To Failure), respectively.

Second, we study failures distribution as Bluetooth channel utilization changes. In other words, we identify usage patterns that have to be avoided to develop more robust applications.

Failure data are gathered from system log files and application logs produced by synthetic workloads, which are appli-

This work has been partially supported by the Italian Ministry for Education, University and Research (MIUR),within the frameworks of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)", and of the PRIN Project "COMMUTA: Mutant hardware/software components for dynamically reconfigurable distributed systems". Authors are grateful to Fabio Cornevilli, Gabriella Carrozza and Giulio Virnicchi for their excellent work on the implementation of the testbed, the integration of the Windows Client, and the coalescence analysis.

cations running on real-world Bluetooth PANs, emulating the behavior of Bluetooth users using different profiles. The reason for emulation is twofold: i) it allows to gather a large amount of data, giving statistical significance to performed measures; and ii) since the workload operates 24/7, it permits the time to failure (TTF), the time to recover (TTR), and related attributes, to be measured. We setup two kinds of heterogeneous testbeds, in two different labs of our department, where field data are collected and then filtered The first testbed stimulates Bluetooth channels via completely random workloads in order to study the Bluetooth Channel behavior irrespective of the specific networked application being used. This let us identify good/bad usage patterns that should be adopted/avoided to realize more robust applications. The second one uses more realistic workloads of traditional IP-based application (i.e., Web, streaming, and Peer-to-peer), adopting the traffic models as published in recent works on this research field. This workload allows to characterize the dependability of traditional networked applications using Bluetooth PAN as a last-meter access network. Both testbeds have been working since one and a half years, collecting more than 300000 failure data items, and they are composed of Linux/Windows commodity PCs and of mobile terminals equipped with different Linux distributions[1].

The rest of the paper is organized as follows. Section 2 briefly presents an overview of the research conducted in the field of dependability of wireless networks. Section 3 describes the Collection Infrastructure we developed. Section 4 describes the failure model of Bluetooth PANs and put the evidence on the effectiveness of the recovery actions and error masking strategies. Section 5 demonstrates how the dependability of Bluetooth PANs can be improved if the previously identified recovery actions and masking strategies are integrated in the Bluetooth Protocol Stack or in the application code. Section 6 discusses further results about failures distribution as a function of several criteria. Section 7 concludes the paper with final remarks and directions of future work.

## 2 Background and Related Research

**Bluetooth.** Bluetooth [3], BT in the following, is a short-range wireless technology operating in the 2.4 GHz ISM band. The BT system provides both point-to-point and point-to-multipoint wireless connections. Two or more units sharing the same channel form a *piconet*. One BT unit acts as the master of the piconet, whereas the other unit(s) acts as slave(s). Up to seven slaves can be active in the piconet. Multiple piconets with overlapping coverage areas form a *scatternet*.

Applications running on BT-enabled devices use a common set of data-link protocols, the BT core protocols, which are described in the following.

*Baseband*: this layer enables the physical RF link between BT units forming a piconet. It provides two different physical links, Synchronous Connection-Oriented (SCO) and Asynchronous Connectionless (ACL). The channel is divided into time slots, each 625 $\mu$s in length. A BT ACL data packet may occupy 1, 3, or 5 consecutive time slots. Packets consist of a 72-bit access code for piconet identification and synchronization, a 18 bit header, and a variable length payload. The payload consists of three segments: a payload header, a payload and a Cyclic Redundancy Code (CRC) for error detection. CRC length is 16 bit, irrespective of the payload size (i.e., from 1 up to 5 slots). In DM$x$ packets (where $x$ is the number of consecutive slots, i.e. 1, 3, and 5), the payload is also coded with shortened Hamming code, in order to perform Forward Error Correction (FEC). DH$x$ packets are instead uncoded. Integrity checks and retransmissions are performed.

*Link Manager Protocol (LMP)*: the LMP is responsible for connection establishment between BT devices . It also provides BT devices with the inquiry/scan procedure.

*Logical Link Control and Adaptation Protocol (L2CAP)*: this layer provides connection-oriented and connectionless data services, including multiplexing capabilities, segmentation/reassembly operations, and group abstractions.

*Service Discovery Protocol (SDP)*: using SDP, device information, services, and characteristics of services can be retrieved.

The BT specification also defines a Host Controller Interface (HCI), which provides developers with an API to access the hardware and to control registers of baseband controller and link manager.

The communication between a BT Host and a Host Controller takes place via either UART or RS232 protocols over serial channels, such as the Universal Serial Bus (USB). Recently, the BlueCore Serial Protocol (BCSP) has been adopted on some devices. It provides a more sophisticated option than its predecessors. BCSP carries a set of parallel information flows between the host and the controller, multiplexing them over a single UART link, and it adds error checking and retransmission.

In this paper focus is on the use of IP over a BT piconet.The BT Special Interest Group defined the PAN profile, that provides support for common networking protocols such as IPv4 and IPv6. The PAN profile exploits the BT Network Encapsulation Protocol (BNEP) to encapsulate IP packets into L2CAP packets and to provide the Ethernet abstraction. When the PAN profile is used, the master/slave switch

[1] A preliminary version of the work, which only encompassed one Linux-based testbed, with the random workload running for six months, was published in [8]. In contrast, this paper considers a more heterogeneous testbed, and it is especially centered on a detailed analysis of eighteen months of failure data.

role operation assumes a key role. A PAN User (PANU) willing to connect to a Network Access Point (NAP) becomes the master since it initiates the connection. As soon as the connection is established at L2CAP level, a switch is performed, because it is important that the NAP remains the master of the piconet in order to handle up to seven PANUs.

**Related Research.** Several studies have been focused on measurement-based analysis, proposing techniques and methodologies for studying error logs collected from a variety of distributed systems. An excellent survey of these methodologies can be found in [14]. Fundamental examples are studies of Network of Workstations [21], Windows operating systems [22], and Large-Scale Heterogeneous Server Environments, such as the one presented in [20]. However, there has been no prior published work on field failure analysis on mobile environments. On the other hand, there are a growing number of works which have been studying dependability issues of wireless infrastructures. A redundant-based technique has been proposed and evaluated in [7] for improving the dependability of 802.11 networks. A similar work has been done in [12], where focus is on WLAN AP Failures. An interesting study on the reliability of ad-hoc wireless network is presented in [5]. A discussion of how the connectivity, coverage and diameter parameters might affect the reliability of the the network has been conducted. In [19] a dependability analysis of GPRS network is presented, based on modeling approaches. Furthermore, in [17], data collection and processing for a wireless telecommunication system have been addressed, and an analysis of failure and recovery rates is discussed. However, the failure data are relative to the fixed core entities (base stations) of a cellular telephone system. Regarding the BT technology, a collection of user-perceived failure data has been performed in [10], in order to give a qualitative failures characterization of BT-enabled devices. The work defines a set of different *test-cases* which have been applied to a variety of BT-enabled devices. Nevertheless, as also authors stated, the results are not purely scientific in that they have no statistical significance.

## 3 Testbed, Workload, and Collection

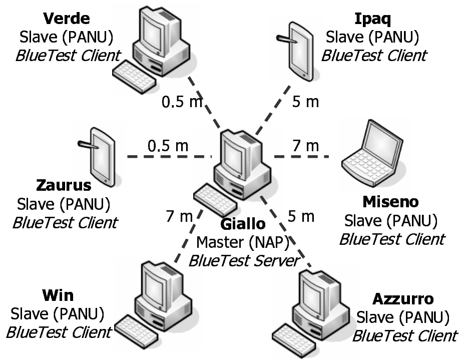
**Testbed Description.** In order to accommodate random and realistic workloads, two testbeds have been deployed, whose topology is shown in figure 1. According to the PAN profile, the master node (Giallo) is configured as a NAP to accept incoming connections from slaves, running PANU applications. The workload, called `BlueTest`, runs on all the nodes, in particular `BlueTest` clients on PANUs, and a `BlueTest` server on the NAP. Both the actual testbeds obey to the same hardware and software configurations, i.e. they are composed of 7 devices, 1 master (the NAP) and 6 slaves (the PANUs). To let results be independent on specific hardware platforms or operating systems, the testbed is composed of heterogeneous nodes, ranging from several commodity PCs, with different hardware configurations and OSs, to PDAs. Table 1 summarizes technical characteristics of the adopted machines. Linux machines use the standard BlueZ stack[2], whereas the Windows ones are equipped with Service Pack 2 and use the BroadCom stack[3]. Note that we could not use the native Windows BT Stack because it does not offer any API for the PAN profile (in Windows XP, IP facilities over Bluetooth are provided only via point-to-point RFCOMM[4] connection). All the machines are equipped with Class 2 BT devices, i.e., up to 10 meters communication range. In order to reduce hardware aging phenomena, the two testbed have been totally replaced by new ones (having the same configuration), in the middle of the testing period. The PANUs' BT antennas have been placed at several different distances from the NAP's antenna (e.g. 0.5m, 5m, and 7m, see figure 1), in order to evaluate the failure distribution as a function of the distance. Antenna positions are fixed, hence we collect data about fixed PAN topologies. However, this is representative of real cases in which PANs are built among computers on a desk.

**Workload Description.** The `BlueTest` workload (WL) emulates the operations that can be made by a real BT user utilizing PAN applications. Each `BlueTest` cycle consists of common BT utilization phases. It first executes an inquiry/scan procedure to discover the other devices in the environment, then it searches the Network Access Point (NAP) service via a SDP Search operation. Once the NAP has been found, the `Bluetest` client connects to it (by creating a BNEP channel on top of a L2CAP connection), switches the role to slave (to let the NAP to be the master of the piconet), uses the wireless link by transfering data to its counterpart (the `Bluetest` server), and finally it disconnects. Before starting a new cycle, the `BlueTest` client waits for a random time, which can be thought as a user passive off time, modeled according to a Pareto distribution, coherently to previous work [9]. To add uncertainty to the piconet evolution, each WL cycle is characterized by several random variables: i) $S$, scan flag, if true, an inquiry/scan operation is performed; ii) $SDP$, service discovery flag, if true, a SDP search is performed; iii) $B$, the Baseband packet type; iv) $N$, the number of packets to be sent/received; v) $L_S/L_R$, the size of sent/received packets; and vi) $T_W$, the passive off waiting time.
$S$ and $SDP$ are introduced since real BT applications do not perform inquiry/scan procedures or SDP searches every time they run. It is possible to exploit caching of the recently discovered devices or services. For each WL cycle,

[2]The Official Linux Bluetooth protocol stack, http://www.bluez.org

[3]Broadcom is a commercial implementation of the BT Stack for Windows, `http://www.broadcom.com`

[4]Serial Cable Emulation Protocol



| Host | O.S. | Distribution | Kernel Version |
|---|---|---|---|
| Giallo | Linux | Mandrake | 2.4.21-0.13mdk |
| Verde | Linux | Mandrake | 2.4.21-0.13mdk |
| Miseno | Linux | Debian | 2.6.5-1-386 |
| Azzurro | Linux | Fedora | 2.6.9-1-667 |
| Win | MS Windows XP | Service Pack 2 | |
| Ipaq H3870 | Linux | Familiar 0.8.1 | 2.4.19-rmk6-pxa1-hh37 |
| Zaurus SL-5600 | Linux | Open Zaurus 3.5.2 | 2.4.18-rmk7-pxa3-embedix |

| Host | CPU/RAM | BT Stack | BT Hardware |
|---|---|---|---|
| Giallo | P4 1.60GHz/128Mb | BlueZ 2.10 | Anycom CC3030 |
| Verde | P3 350Mhz/256Mb | BlueZ 2.10 | 3COM 3CREB96B |
| Miseno | Celeron 700Mhz/128Mb | BlueZ 2.10 | Belkin F8T003 |
| Azzurro | P3 350MHz/256Mb | BlueZ 2.10 | Digicom Palladio |
| Win | P4 1.80Ghz/512Mb | Broadcomm | Sitecom CN-500 |
| Ipaq H3870 | StrongARM 206 MHz/64Mb | BlueZ 2.10 | *on board* |
| Zaurus SL-5600 | XScale 400Mhz/32Mb | BlueZ 2.10 | *on board* |

**Figure 1.** The topology of both the testbeds, along with the technical details of their machines

values for $S$ and $SDP$ are chosen according to the uniform distribution, since the lack of publicly available information about the real utilization pattern of inquiry/scan procedures and SDP searches for a typical PANU application.
$B$, $N$, $L_S$, and $L_R$ parameters depend on the channel utilization, as described in the following.

**Random WL.** It generates totally random values for $B$, $N$, $L_S$, and $L_R$. In particular, $B$ is randomly chosen among the six BT packet types (i.e. DM$x$ or DH$x$), according to a binomial distribution. This helps to 'stimulate' the channel with every packet types. $N$, $L_S$, and $L_R$ are generated following uniform distributions, for the same reasons. More details on this WL can be found in our previous work [8].

**Realistic WL.** It generates values for the parameters according to the random processes which are used to model actual Internet traffic [9, 11]. In particular, the choice for $B$ is left to the BT Stack, whereas $N$ follows power law distributions (e.g. the Pareto distribution) related to the dimension of the resource that have to be transferred. The parameters of the distributions are set with respect to the application being emulated (e.g. Web browsing, file transfer , e-mail, peer to peer, video and audio streaming). Values for $L_S$ and $L_R$ are set according to the actual Protocol Data Unit commonly adopted for the various transport protocols over the Internet [11]. Finally, since a user can run more applications in sequence over the same connection, the WL runs from 1 up to 20 consecutive cycles over the same connection. Further details on the Realistic WL can be found in [6].

**Failure Data Sources and Collection Methodology.** Failures might manifest during the normal WL execution. When a failure occurs, the workload is instrumented to register a failure report, as will be detailed later. Two levels of failure data are produced, as in the following.
*User Level Failures or High Level Data*: failure reports about the failure as it manifests to a real user, using a PANU device. The report also contains details about the BT node status during the failure (e.g. the WL type, the packet type, the number of sent/received packets);
*System Level Failures or Low Level Data*: failure data registered by system software on the OS system log file, including BT APIs and OS drivers.
System Level Failures can be seen as errors for User Level Failures. In other terms, when a User Level Failure manifests, one or more System Level Failures are registered in the same period of time. This helps to understand causes behind the high level manifestation. Failure data on each BT node is collected by a `LogAnalyzer` daemon, and is sent to a central repository, where data are then analyzed by means of a statistical analysis software. We used the SAS analyzer suite[5].
On each BT node, both User Level and System Level failure data is stored in two files: the *Test Log file*, which contains User Level failures reports, and the *System Log file*, containing all the error information registered by the applications and system daemons running on the BT host machine. The `LogAnalyzer` periodically i) extracts failure data from both the logs, ii) filters them, and iii) sends them to the repository. Filtering is used to send only significant data to the repository. At the time of writing, both testbeds have been running for more than one year, from June 2004 to November 2005, with 356551 failure data items being collected. In particular, there were 20854 User Level failure reports from Test Logs and 335697 System level failure entries from System Logs. The most of the failures (the 84%) comes from the random workload (see section 6).

## 4 Key Findings

**Bluetooth PAN Failure Model.** Failures are classified by analyzing their spontaneous manifestation, as recorded

[5]SAS is an integrated software platform for business intelligence which includes several tools for statistical analysis; it is produced by SAS Institute Inc., `http://www.sas.com`

| Bluetooth PAN Failure Model | | | | | |
|---|---|---|---|---|---|
| User Level Failures | | | System Level Failures | | |
| Group | Type | Phenomenology | Location | Type | Observed errors |
| SDP Search | Inquiry/Scan failed | The inquiry procedure terminates abnormally. | BT Stack related | HCI | Command for unknown connection handle, timeout in the transmission of the command to the BT firmware. |
| | NAP not found | The SDP procedure does not find the NAP, even if it is present. | | | |
| | SDP Search failed | The SDP Search procedure terminates abnormally. | | SDP | Connection with the SDP server refused or timed out, AP unavailable or not implementing the required service, even if it implements it. |
| Connect | Connect failed | The device fails to establish the L2CAP connection with the NAP. | | | |
| | PAN connect failed | The PANU fails to establish the PAN connection with the NAP. | | L2CAP | Unexpected start or continuation frames received. |
| | Bind failed | The IP socket cannot bind the Bluetooh BNEP interface. | | BNEP | Failed to add a connection, can't locate module bnep0, bnep occupied |
| | Switch role request failed | The switch role request does not reach the master. | OS, Drivers related | USB | The USB device does not accept new addresses to communicate with the BT hardware. |
| | Switch role command failed | The request succeeds, but the command completes abnormally. | | | |
| Data Transfer | Packet loss | An expected packet is lost, since a timeout (set to 30 secs) expires. | | BCSP | Out of order or missing BCSP packets. |
| | Data mismatch | The packet is received correctly, but the data content is corrupted. | | Hotplug timeout | The Hardware Abstraction Layer (HAL) daemon times out waiting an hotplug event. |

**Table 1.** Bluetooth PAN Failure Model

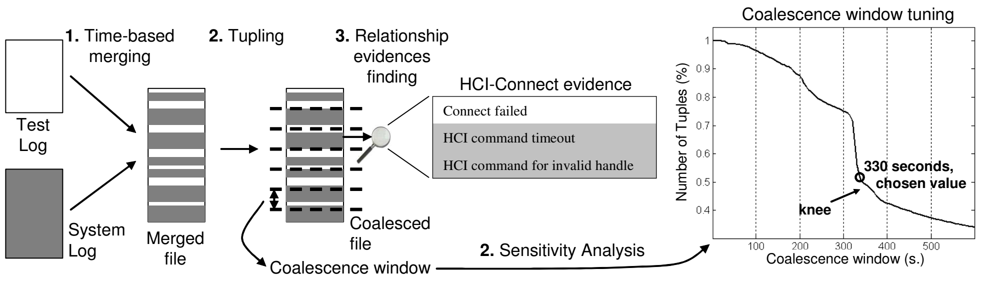


**Figure 2.** The "merge and coalesce" scheme adopted to pinpoint error-failure relationships.

in our Test and System logs. The failure model described here is general, in the sense that it considers all failure reports and events gathered from both testbeds and all the machines. Failure manifestations are workload independent, i.e., the same failure types have been observed, regardless of the workload being run. Differences are in the failure rates, as will be detailed in section 6.

Table 1 gives an overall picture of the Bluetooth PAN failure model. This considers two levels of failures, user and system level, according to the collection methodology previously described. The reported failure types are the result of an accurate classification of the collected user failures' reports. Failure messages related to the same failure have been classified into one failure type. This gives the model simplicity and understandability[6]. Left-side columns in table 1 describe user level failures, whereas right-side columns are dedicated to system level failures. User level failures can be grouped into three classes, in accordance with the utilization phase in which they manifest, i.e., searching for devices and services, connecting, and transferring data. Each group contains one or more failure types. For each failure type, a brief description of its phenomenology is given in the table. Unexpectedly, failures during data transfer ('Data Transfer' group) encompass packet losses and data corruption, despite Baseband's error control mechanisms, such as CRCs, and FEC schemes. However, as discussed in previous work [18], the weakness of integrity checks is the assumption of having memoryless channels with uncorrelated errors from bit to bit. In our case, correlated errors (e.g. bursts) can occur due to the nature of the wireless media, affected by multi-path fading and electromagnetic interferences.

System level failures are grouped with respect to their location, i.e., BT software stack and OS/Drivers. Failure types have been defined according to the component which signaled the failure. For each failure type, a brief description is given.

To gain more insights into error-failure relationship, we relate User Level with System Level failures. Due to the huge amount of data, this operation cannot be performed manually. Hence, we define a novel "merge and coalesce" semi-automatic scheme, whose steps are summarized in figure 2. First, for each node a log file is produced by merging its `Test Log` and `System Log` files, on a time-based crite-

[6]Unclassified user level failures can be found on the web site of the project: `http://www.mobilab.unina.it/BTDepend.htm`.

|  | System Level Failures | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **User Level Failures** | L2CAP | | HCI | | SDP | | BNEP | | HOTPLUG | | USB | | BCSP | | |
| **From** | local | NAP | local | NAP | local | NAP | local | NAP | local | NAP | local | NAP | local | NAP | **TOT** |
| Inquiry/scan failed | | | | | | | | | | | | | | | 0.1 |
| Nap not found | | | 18.8 | | 61 | | | | | | | | 20.2 | | 0.2 |
| SDP search failed | | | 20 | | 50.9 | | | | | | | 9.1 | 20 | | 0.1 |
| Connect failed | | 4.9 | 85.1 | 2.3 | | | | | | | 1 | 1.5 | 5.2 | | 5.7 |
| PAN connect failed | | | | | 96.5 | | 3.5 | | | | | | | | 0.5 |
| Bind failed | | 3.6 | 55.5 | 1.8 | | | | | 35.5 | | | 3.6 | | | 38.6 |
| Sw role command failed | 0.9 | 4.4 | 10.9 | 2.4 | 8.2 | | 18.8 | | | | 2.7 | 2 | 49.7 | | 19.4 |
| Sw role request failed | | | 91.1 | | | | | | | | | | 8.9 | | 0.8 |
| Packet loss | 2.7 | 3.9 | 21.8 | 2.5 | 0.9 | | 15.4 | | 17.2 | | 0.9 | 2.6 | 32.1 | | 33.9 |
| Data mismatch | | | | | | | | | | | | | | | 0.7 |
| **Total** | 1 | | 49.9 | | 7 | | 8.5 | | 11.4 | | 1.1 | | 21.1 | | |

**Table 2.** Error-Failure Relationship: System Failures are regarded as errors for User Failures.

ria (entries are ordered according to their timestamps). Second, the merged file is analyzed using the tupling coalescence scheme [4], i.e., if two or more events are clustered in time, they are grouped into a tuple, according to a coalescence window. The window size has been determined by conducting a sensitivity analysis, as shown in the plot in figure 2. The number of obtained tuples (reported as percentage of entries on the vertical axis) is plotted as a function of the window size. A critical "knee" is highlighted in the plot. Choosing a point on the curve before the knee causes the number of tuples to drastically increase, thus generating truncations, i.e., events related to the same error are grouped into more than one tuple. On the other hand, choosing a point after the knee generates collapses because events related to different errors are grouped into the same tuple, due to the decrease of the number of tuples. For these reasons, a window size equals to 330 seconds, that is, exactly at the beginning of the knee, is chosen. Third, the error-failure relationship is inferred by analyzing tuples' contents. For instance, if a tuple contains both a *Connect failed* high level message, and HCI low level messages, an evidence of a HCI-connect relationship is found. Counting all the HCI-connect evidences gives a mean to weight the relationship.
Table 2 illustrates the results obtained by applying the mentioned approach. The interpretation of the table is simple: the greater is the percentage reported in a cell, the stronger is the relationship between the user level failure (on the row) and the system level failure (on the column). Percentages on each row sum to 100 (except from the "tot" column), so as to have a clear indication of user level failure causes. The "total" and "tot" report the total percentages, e.g., the 49.9% of the user failures are due to HCI system failures. In order to discover error propagation phenomena from the NAP to PANUs, the user level data have also been related with the NAP's system log file (i.e., the server), with the same "merge and coalesce" approach. Hence, for each system level failure column, the table reports the figures obtained by relating the Test log with both the local system log and the NAP system log, for each machine. The table contains very useful information about the error-failure relationship, and NAP-PANU propagation phenomena. For example, failures during the L2CAP connection (row *Connect failed* in the table) are mostly due to timeout problems in the HCI module, either from the local machine or from the NAP. This occurs when a connection request (or accept) is issued on a busy device. PAN connection failures are instead frequently related to failures reported by the SDP daemon (the 96.5% of the cases). Interestingly, we observed that exactly the 96.5% of PAN connect failures manifests when the SDP Search is not performed by the workload (in other terms, when the $SDP$ flag is false). This is a clear indication that *avoiding caching and performing the SDP search before a PAN connection is a good practice to reduce PAN connect failures occurrence*. One more interesting relationship is between "Switch role request failed" and command transmission timeouts signaled by the HCI module (the 91.1% of switch role request failures). This suggests that *increasing the timeout in the API helps to reduce the switcth role request failure occurrence*. For some failures, such as "Inquiry/Scan failed", no relationships has been found. Table 2 also permits to define masking strategies, as detailed in next sub-section.

**Error Masking Strategies.** Error masking strategies have been defined for the following user level failures.
*Bind failed*: it is mostly related to problems in the host OS hotplug interface or to errors when invoking HCI commands (see table 2). Our investigation on source code on BT Kernel modules has led to the following considerations. The creation process of an IP network interface over BT requires: i) a time interval ($T_C$) for the creation of the

L2CAP connection; and ii) a time interval ($T_H$) needed by the BT stack to build the BNEP virtual network interface over L2CAP, and by the OS hotplug interface to configure the interface. The problem is that the Pan Connect API is not synchronous with $T_C$ and $T_H$, hence, a "bind failed" failure occurs whenever the application attempts to bind a socket on the supposed existing BNEP interface before $T_C$ and $T_H$. In particular, if the bind request is issued before $T_C$, a HCI command failure (i.e., command for invalid handle) occurs, because the L2CAP connection is not present. If the request is instead issued after $T_C$ but before $T_H$, a failure occurs, either because the interface is not present or it does not have been configured yet by the hotplug mechanism. To prevent the failure from occurring, it is sufficient to wait for $T_C$ and $T_H$ to elapse. $T_C$ elapses as soon as the L2CAP connection has a valid handle. This check can be easily added in the PAN connection API. As for $T_H$, the OS hotplug interface can be instrumented so as to notify the application as soon as the BNEP interface is up and configured.
*Switch role command failed* and *NAP not found*: the switch role failure is often related to out of order packets failures signaled by the BCSP module (49.7% of the cases, see table 2). Also, it especially manifests on PDAs (see section 6), since they adopt BSCP. The failure can also be related to many other causes, such as unexpected L2CAP frames (0.9% local, 4.4% on the NAP), HCI command for invalid handle (10.9% local, 2.4% on the NAP), and occupied BNEP device(18.8% local). This multitude of transient causes does not isolate the symptoms of the failure, and it does not allow to define precise maskings. Therefore we tried to simply repeat the command when it fails. We experienced that repeating the action up to 2 times (with 1 second wait between a retry and the successive) is enough to let the underneath transient cause disappear, and hence to make the command success. The same considerations apply to the case of NAP not found.

**SW Implemented Recovery Actions.** As soon as a failure is detected[7], several Software Implemented Recovery Actions (SIRAs) are attempted in cascade. This approach allows to pinpoint, for each failure, the most effective recovery action, that is, the one that fixes the problem with a high probability. Note that this is the only viable approach, since we do not have any *a priori* knowledge about the best recovery to perform each time. Upon failure detection, the following recovery actions are triggered subsequently, i.e., when the $i$-th action does not succeed, the $(i+1)$-th action is performed.
*1. IP socket reset*: the socket is destroyed and then rebuilt;

[7]Failure detection is performed by simply checking the return state of each BT or IP API that is invoked by the WL. Examples are the indication that a PAN connection cannot be created, or a timeout when waiting for an expected packet.

| | SIRAs | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **User Level Failures** | IP socket reset | BT connection reset | BT stack reset | Application restart | System reboot | Multiple app restart | Multiple sys reboot | **TOT** |
| Inquiry/scan failed | | | 34.5 | 30 | | 35.5 | | 0.1 |
| Nap not found | | | 61.4 | 3.9 | 30.8 | 3.9 | | 0.2 |
| SDP search failed | | 40.1 | 39.8 | | 20.1 | | | 0.1 |
| Connect failed | | 0.5 | 14.9 | 55.8 | 25.6 | 0.1 | 3.1 | 5.7 |
| PAN connect failed | | 46.4 | 35.7 | 5.4 | 12.5 | | | 0.5 |
| Bind failed | 0.1 | 5.5 | 62.4 | 30 | 1.7 | | 0.3 | 38.6 |
| Sw role command failed | | 63.7 | 20.4 | 11.3 | 2.4 | 1.8 | 0.4 | 19.4 |
| Sw role request failed | | 28.4 | 48.2 | 4.9 | 17.3 | 1.2 | | 0.8 |
| Packet loss | 5.9 | 7.2 | 25.8 | 33.1 | 26.7 | 0.2 | 1.1 | 33.9 |
| Data mismatch | | | | | | | | 0.7 |
| **Total** | 2 | 17.5 | 38.9 | 28.4 | 12 | 0.5 | 0.7 | |

**Table 3.** User failures-SIRA relationship.

*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, by restoring the initial states;
*4. Application restart*: the `BlueTest` is automatically closed and restarted;
*5. Multiple application restart*: up to 3 application restart are attempted, consecutively;
*6. System reboot*: the entire system is rebooted;
*7. Multiple system reboot*: up to 5 system reboot are attempted.
Note that the defined SIRAs have also been performed in order to keep the testbed up. The given recovery actions are ordered according to their increasing costs, in terms of the recovery time. The more attempts have to be done for a failure, the more the failure is severe: if action $j$ was successful, we can say the failure has a severity $j$. This gives us an indication for failure severity.

Table 3 highlights the relationship between user level failures and recovery actions. Data is relative to all the machines and come from both testbeds, and does not take into account masking strategies. Each number in a cell represents the percentage of success of the recovery action (on the column) with respect to the given user level failure (on the row). Therefore, the figures give an indication of the effectiveness of each SIRA for each failure (which is an estimation of the probability that a certain recovery action goes through). Similarly to table 2, numbers on each row sum to 100, in order to have a simple indication of which are the effective SIRAs for each user failure. For example, a "NAP not found" is most probably recovered by resetting the BT stack (in the 61.4% of the cases). Hence, this should be the first action to be attempted when the failure is detected. The table also allows to calculate failure severity. For instance, the "Connect failure" is one of the most severe, since it is often recovered by expensive SIRAs (the 84.6% of the cases from "Application restart" up to "Multiple system reboot"). Finally, failure-recovery relationship provides further understanding of failure causes. As an example, packet losses recovered by an IP socket reset (the 5.9% of packet losses) may be due to interferences over the wireless media. It is indeed not necessary to reestablish the L2CAP and PAN 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. For "data mismatch" failures, no recoveries are defined. Data mismatch is not realistically recoverable, since a real application only relies on integrity mechanisms furnished by the communication protocols, and cannot know the actual instance of data being transferred.

## 5 Dependability Improvement

From collected data, and from results of the previous section, it is possible to estimate the dependability improvement which can be obtained by integrating software implemented recovery actions and error masking strategies into the testbeds. To this aim, we consider two typical usage scenarios: i) each time that a failure occurs, a typical user performs the reboot of the terminal (PC or a PDA); ii) the user performs the following recovery actions, subsequently, ii.1) he/she tries to restart the application, and ii.2)in the case that the application fails again, he/she reboots the terminal. For both scenarios we are able to evaluate Mean TTF and TTR values (MTTF and MTTR), since we can calculate the average recovery time for reboot and for the application restart from the collected data. In order to obtain upper bound measures, we assume that the 'user thinking time' is zero, i.e. we do not encompass it in the TTR value. Finally, we compare the two scenarios with the enhanced facilities that we implemented in the workload, i.e. software implemented recovery actions and error masking. Results are summarized in table 4.

As for the coverage, we refer to failure mode coverage as defined in [1](*failure assumptions coverage*). The testbed with the automated recovery actions gives a coverage of 58.4%, i.e. it is able to recover 58.4% of failures without rebooting the system or restarting the application (as a typical user would have done). In the fourth column of the table we reported results taking also into account the error masking strategies, which gives a coverage of 73.61%. This, in our opinion, represents a good result for the effectiveness of fault tolerance techniques we have found from the analysis of gathered data. As far as the availability is concerned, results show that the software implemented recovery actions and the error masking strategies actually improved the availability of BT PANs, starting from 0.688

| | Only Reboot | App restart and Reboot | With only SIRAs | SIRAs and masking |
|---|---|---|---|---|
| **MTTF (s.)** | 630.56 | 831.38 | 845.54 | **1905.05** |
| $MIN_{TTF}$ (s.) | 11 | 11 | 11 | 19 |
| $MAX_{TTF}$ (s.) | 117893 | 117893 | 117893 | 117893 |
| $DEV\_STD_{TTF}$ (s.) | 2833.05 | 2984.12 | 2997.36 | 5311.7 |
| **MTTR (s.)** | 285.92 | 85.1 | **70.94** | **120.84** |
| $MIN_{TTR}$ (s.) | 210 | 10 | 2 | 2 |
| $MAX_{TTR}$ (s.) | 7366 | 7366 | 7366 | 7366 |
| $DEV\_STD_{TTR}$ (s.) | 263.71 | 112.64 | 99.4 | 128.17 |
| **Availability*** | < 0.688 | < 0.907 | **0.923** | **0.94** |
| **% Coverage** | 0 | 0 | 58.4 | **73.61**** |
| **% Masking** | 0 | 0 | 0 | 58 |

*= MTTF/(MTTF+MTTR)

** = 58% (masking) + 15.61% (coverage of the remaining failures)

**Table 4.** Dependability Improvement

(scenario 1) and 0.907 (scenario 2), which are the upper bound measured values of BT PAN availability, to 0.923 and 0.94, with an improvement of 3.64% (relative to scenario 2), up to 36.6% (relative to scenario 1). The error masking strategies influence the MTTF estimation, which varied from 630 $s$. to 1905 $s$. This results in an actual reliability improvement of 202%. It should be noted that, even using SIRAs and masking, the MTTF values are low, i.e., each 30 minutes on average a node in the piconet fails. Since our measurements are based on a 24/7 experiment, this represents a major reliability issue in all those scenarios in which piconets are permanently deployed and used continuosly, such as, wireless remote control systems for robots, and aircraft maintenance systems. In these critical scenarios, extensive fault tolerance techniques shoud be adopted, such as, using redundand, overlapped piconets, other than SIRAs and masking. Finally, as for the maintainability, the MTTR decreases from 285.92 $s$. to 70.94, thanks to SIRAs. It results instead slightly longer (120.84 $s$.) in the case with both SIRAs and masking, since the remaining, unmasked failures (the 42% of the failure) are generally severe, and require costly recovery actions.

## 6 Failure Distributions

Figure 3a shows how packet losses distribute with respect to the Baseband packet type. Data are from the random WL. Results evidence that, from a dependability point of view, it is better to *use multi-slot packets*. Another suggestion is to *prefer DHx packets to DMx ones*. Both phenomena have the following explanation. Retransmissions at the Baseband level are allowed up to a certain limit at which the current payload is dropped and the next payload is considered [3]. Hence, more packets are dropped in the case of

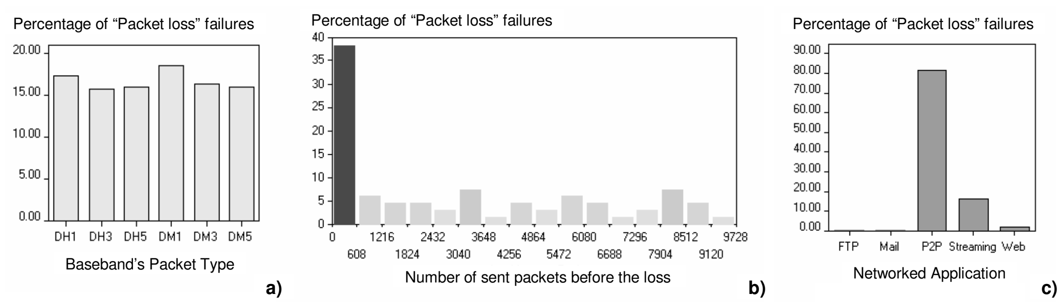


**Figure 3.** "Packet loss" failure distribution as a function of a) the packet type, b) the number of sent packets before the loss, and c) the used networked application

strict error control (i.e., DM$x$ and single slot packets), due to more retransmissions.

A further finding is to *keep an already open connection up, instead of closing it as soon as no more data has to be sent.* To justify this claim, we demonstrate that i) connections which are short in length fail more; by length we mean the number of packets sent before that the failure occurs, and ii) leaving a connection up and unused (i.e., idle connection) does not cause more failures to occur. To answer point i), an experiment has been performed with a special version of the random WL. The experiment has been run for two months on two machines (Verde and Win). This WL has $N$ fixed to 10000 packets, and both $L_R$ and $L_S$ fixed to 1691 bytes (that is, the BNEP MTU). These parameters are fixed in order to not introduce indetermination when estimating the failing connection length. Results are shown in figure 3b, where packet loss failures distribution is reported as a function of the number of sent packets prior to the failure. They demonstrate that it is more frequent for a connection to fail when it is young. This is likely due to latent errors of the connection setup process, such as the corruption of the BT stack data structures. To answer point ii), we looked at the data from the realistic WL, which runs up to 20 consecutive cycles on the same connection, keeping the connection idle for a time $T_W$ between two cycles. We discovered that the average $T_W$ obtained considering only idle times preceding a failed cycle (27.3 s.) is almost equal to the average $T_W$ when considering only idle times before failures-free cycles (26.9 s.)[8]. This is an evidence that idle connections do not cause more failures to occur.

The previous claim is also supported by looking at the difference between our WLs: the Random WL creates and destroys connections frequently, whereas the realistic WL exploits the established connections more. As a result, the former generates more failures (84%) than the latter (16%).

[8]The measure discards the idle times relative to consecutive cycles on different connections. We recall that the $T_W$ follows a Pareto distribution with a shape parameter equal to 1.5, in coherence with previous work [9].

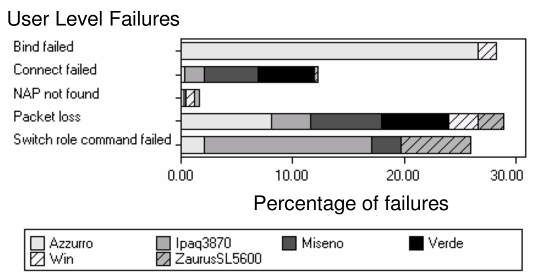


**Figure 4.** User failures-nodes relationship

Figure 3c shows the "Packet loss" distribution as a function of the networked application that was run by the WL during the failure. Data is relative to the Realistic WL testbed. Results pinpoint Peer to Peer (P2P) and Streaming applications as the most critical for BT PANs. They are indeed characterized by long sessions with continuous data transfer, which overload the channel and stress its time-based synchronization mechanism. At a first glance, this may surprise, since P2P protocols are TCP-based. However, the most of the packet losses are due to broken BT links, which cause the TCP end-to-end channel to brake as well. Streaming causes less failures than P2P due to its isochronous nature, which better fits the BT time-based nature. Less failures are experienced with Web, Mail, and File Transfer Protocol (FTP) applications, which are characterized by intermittent transfers. This indicates that *Bluetooth ACL channels are less failure prone when used in an intermittent manner*.

Figure 4a depicts the frequency distribution of the most significant user level failures as a function of the host, when no masking strategies are applied. *Giallo* is not present, since the NAP only records system level data. Results are obtained only from the Realistic WL, thus the failure rates are different to those shown in table 2. It should be noted that bind failures only appeared on *Azzurro* and *Win*. On *Azzurro*, that runs a Fedora Core distribution, the problem remained even after upgrading the hardware to a Pentium 4

1.8 GHz with 512 Gb RAM. The problem is hence probably due to the new version of the Hardware Abstraction Layer (HAL), firstly deployed on fedora core distributions, and responsible for the hotplug mechanism. “Switch role command” failures are frequent on PDAs, due to the complexity introduced by the BCSP. Finally, it is interesting to notice that *the failure distribution is not significantly influenced by the distance* between the BT antennas. From data relative to the Realistic WL we measured that the 33.33%, 37.14%, and 29.63% of failures occur with a distance 0.5 $m$, 5 $m$, and 7 $m$, respectively. Bind failures are not taken into account in the count. They would have biased the measure, since they only manifest on two hosts.

## 7 Conclusions and Future Work

This paper presented a field failure data analysis of Bluetooth PANs. Presented results have shown how failure data provide helpful insights to design fault tolerance means for operational systems. Respectively, up to 36,6% and 202% availability and reliabilty improvements have been demonstrated, by defining and using SIRAs and error masking strategies. Several lessons have also been learned about preferable usage patterns, from a dependability perspective. Examples are to avoid caching by performing the SDP search before the PAN connection, to adopt multi-slot, DH$x$ packets, to use long lived connections, and to increase the timeout in the switch role API. The insights are gained from the definition of a failure model for BT PANs, completed with error-failure relationships, that researches can use to design abstract models useful for further analysis or synthesis. At time of this writing we are carrying out an enhanced version of the Linux BlueZ BT protocol stack, which includes all the findings we gathered from the analysis, and that developers can use for building more robust BT applications.

## References

[1] A.Avizienis, J. Laprie, B.Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. on Dependable and Secure Computing*, 1(1):11–33, January-March 2004.

[2] J. E. Bardram. Applications of context-aware computing in hospital work-examples and design principles. *Proc. of the 19th ACM Symposium on Applied Computing (SAC 2004)*, March 2004.

[3] Bluetooth SIG. *Specification of the Bluetooth System - core and profiles v. 1.1*, 2001.

[4] 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.

[5] S. Cabuk, N.Mahlotra, L. Lin, S. Bagchi, and N. Shroff. Analysis and evaluation of topological and application characteristics of unreliable mobile wireless ad-hoc network. *proc. of 10th IEEE Pacific Rim International Symposium on Dependable Computing*, 2004.

[6] G. Carrozza, M. Cinque, F. Cornevilli, D. Cotroneo, C. Pirro, and S. Russo. Architecting a Realistic Workload for Bluetooth PANs Stressing. TR-WEBMINDS-58, University of Naples Federico II, web-minds.consorzio-cini.it, November 2005.

[7] D. Chen, S. Garg, C. Kintala, and K. S. Trivedi. Dependability Enhancement for IEEE 802.11 with Redundancy Techniques. *proc. of IEEE 2003 International Conference on Dependable Systems and Networks (DSN ’03)*, June 2003.

[8] M. Cinque, F. Cornevilli, D. Cotroneo, and S. Russo. An Automated Distributed Infrastructure for Collecting Bluetooth Field Failure Data. *Proc. of the 8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC 2005)*, May 2005.

[9] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *Proc. of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 1996.

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

[11] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. Diot. Packet-level Traffic Measurements from the Sprint IP Backbone. *IEEE Network*, 17(6):6–16, December 2003.

[12] R. Gandhi. Tolerance to access-point failures in dependable wireless lan. *Proc. of the 9th Int. Workshop on Object-Oriented Real-Time dependable Systems (WORDS’03)*, June 2003.

[13] M. Gerla, P. Johanssona, R. Kapoor, and F. Vatalaro. Bluetooth: ”last meter” technology for nomadic wireless internetting. *Proc. of 12 th Tyrhennian Int. Workshop on Digital Communications*, 2000.

[14] 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, Springer Verlag*, 1999.

[15] 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.

[16] M. Lampe, M. Strassner, and E. Fleisch. A Ubiquitous Computing Environment for Aircraft Maintenance. *Proc. of the 19th ACM Symposium on Applied Computing (SAC 2004)*, March 2004.

[17] 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)*, 2002.

[18] M. Paulitsch, J. Morris, B. Hall, K. Driscoll, E. Latronico, and P. Koopman. Coverage and the Use of Cyclic Redundancy Codes in Ultra-Dependable Systems. *Proc. of the IEEE International Conference on Dependable Systems and Networks (DSN 2005)*, June 2005.

[19] S. Porcarelli, F. D. Giandomenico, A. Bondavalli, M. Barbera, and I. Mura. Service-level availability estimation of gprs. *IEEE Transactions on Mobile Computing*, 2(3), July-September 2003.

[20] R. K. Sahoo, A. Sivasubramaniam, M. S. Squillante, and Y. Zhang. Failure data analysis of a large-scale heterogeneous server environment. *proc. of the 2004 International Conference on Dependable Systems and Networks (DSN’04)*, June 2004.

[21] A. Thakur and R. K. Iyer. Analyze-now - an environment for collection and analysis of failures in a networked of workstations. *IEEE Transactions on Reliability*, 45(4):560–570, 1996.

[22] J. Xu, Z. Kalbarczyc, and R. K. Iyer. Networked Windows NT System Field Data Analysis. *proc.of IEEE Pacific Rim International Symposium on Dependable Computing*, December 1999.