



# Distributed Real-time Architecture for Mixed Criticality Systems

## *Assessment report for mixed-criticality healthcare and entertainment use cases*

## *D 8.3.2*

| | | | | | |
|---|---|---|---|---|---|
| **Project Acronym** | DREAMS | **Grant Agreement Number** | | FP7-ICT-2013.3.4-610640 | |
| **Document Version** | 1.0 | **Date** | 2017-07-21 | **Deliverable No.** | D 8.3.2 |
| **Contact Person** | Kevin Chappuis | **Organisation** | | Virtual Open Systems | |
| **Phone** | +33663254852 | **E-Mail** | | k.chappuis@virtualopensystems.com | |

# Contributors

| Name | Partner |
|---|---|
| Kevin Chappuis | VOSYS |
| Jeremy Fanguède | VOSYS |
| Miltos Grammatikakis | TEI |
| Marcello Coppola | ST |
| Jörn Migge | RTAW |
| Javier Coronel | FENTISS |
| | |
| | |

# Table of Contents

# 1 Introduction

This document is the deliverable D8.3.2 of the DREAMS project. It is the last deliverable of task *T8.3 – Project technologies assessment* of work package *WP8 – Healthcare Use Case and Demonstrator*. This deliverable, *D8.3.2 – Assessment report for mixed-criticality healthcare and entertainment use cases,* describes the final assessment of the demonstrator for mixed-criticality healthcare and entertainment.

In this document, the healthcare demonstrator is presented, as well as the assessments of the DREAMS objectives related to the WP8 demonstrator.

## 1.1 Position of the Deliverable in the Project

This deliverable relates to task T8.3. Over the course of the project, the task provides two deliverables, both of which aim to assess the technological results of the project related to the healthcare demonstrator along with its mixed-criticality use cases. The confidentiality level of this deliverable is public (PU) and it will be published on the DREAMS website, once approved by the European Commission.

A previous deliverable, D8.2.1 [1], delivered in month 45, provided a general overview of the hardware and software architecture of the demonstrator, while this deliverable evaluates the DREAMS technologies involved in the mixed-criticality healthcare and entertainment demonstrator. A preliminary assessment was performed in deliverable D8.3.1 [2], delivered in month 30, while this deliverable is the final assessment report.

## 1.2 Contents of the Deliverable

In chapter 2, we provide the description of the Healthcare demonstrator platform and the DREAMS technologies used. In chapter 0 the evaluation methodology as well as the assessment of the results of the demonstrator is presented.

# 2 WP8 Healthcare Demonstrator: Platform, Technologies and Application

## 2.1 Hospital Use Case

The demonstrator is implementing the use-case of a Hospital Media Gateway, where the rooms are connected to provide medical information to the hospital staff but also to provide entertainment media to patients. Thus, the system demonstrator running mixed-criticality healthcare and entertainment use case involves the consumption of media content by several patients located in different Hospital rooms, while monitoring of the overall health is performed using the ST Body Gateway devices. The Healthcare demonstrator is a distributed system composed of several devices that are communicating with a central Hospital Server. This server provides the on-demand media content consumption by the TV sets (using a wired network) without the need of a dedicated per-room set-top box (STB). In addition, the Hospital server is used to store the continuous flow of physiological info and it also detects any abnormality arising from a range of medical conditions. When the patient has a clinically relevant event, the Body Gateway communicates via the room wireless switch to the Hospital server. This information is automatically examined by the smart caregiver application in the hospital server and if a critical condition is detected a warning is raised to the hospital personnel for the necessary intervention. At the same time, this technology can provide a valuable history feedback on the condition of the person wearing the body monitoring system. Patients' data logged on the server can be analysed via a dashboard executed on the Hospital server and Patients' data can be queried on demand.

The smart caregiver application will be executed in the secure and critical environment where real-time properties and uptime must be guaranteed while other non-critical applications such as the on-demand content consumption are concurrently executed in a separate environment, much like what is happening today in modern car for the infotainment system.

## 2.2 Demonstrator Overview

The demonstrator platform represents an implementation of the Hospital use case at a reduced size. Figure 1, below, shows the network architecture of the demonstrator. The Body Gateway Control Units that are responsible for collecting all the sensor data, which are, then, relayed to the distribution network via a room gateway. A wired network is responsible for the communication between the room gateway and the Hospital Server. The patient data are collected in a prompt and reliable way to the Hospital server and several incoming media requests are delivered to the Hospital Server. Body Gateways (BGW) are connected via Bluetooth, while media viewers are connected via Ethernet. Then the main communication backbone is implemented using the TTEthernet that enables the native support of different traffic classes having different criticality levels. In order to be scalable, the backbone includes several TTEthernet switches, however in the demonstrator we just use one single switch.

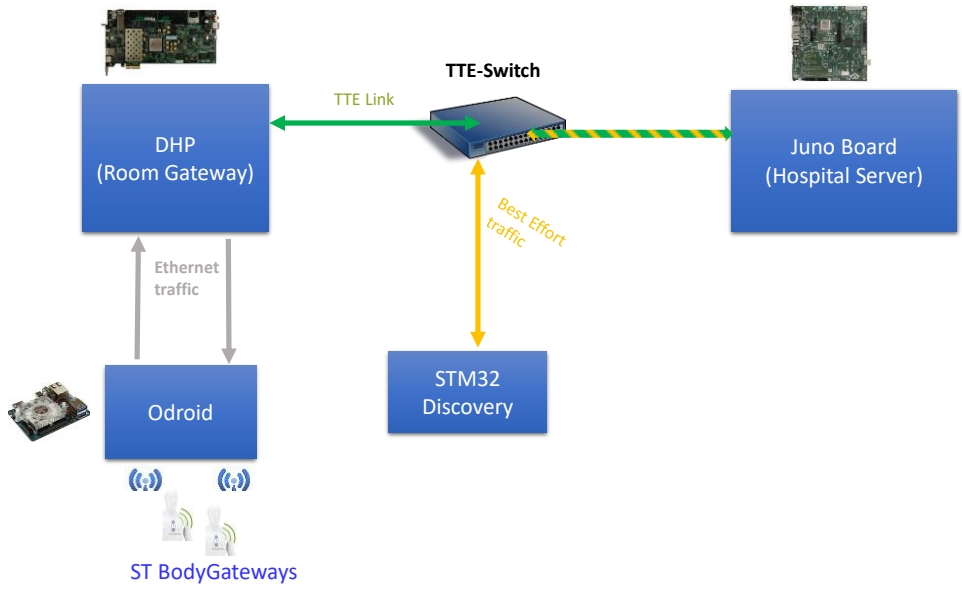


**Figure 1: Network architecture of the demonstrator**

The Hospital server is implemented using the ARM JUNO platform that includes the 64bit ARM Cortex A53 and A57 processors where DREAMS technologies and services are implemented. The room gateway is implemented via the Dreams Harmonized Platform (DHP) that in real life is responsible to provide the wired and wireless access points in each room. A wireless access point is used by the Body Gateways that have been paired, while a wired access point is used by the on-demand media devices. In order to avoid to spend a lot of effort to implement the Bluetooth protocol stack in the DHP board, we have decided to add a simple BT Odroid XU4 board and to add directly the on-demand media devices to the TTEthernet switch. The BT Odroid includes already the Bluetooth network stack enabling to pair directly several body gateways. Then the BT Odroid is connected via Ethernet to the DHP board. Figure 2, below, presents a photo of the demonstrator including all its components.

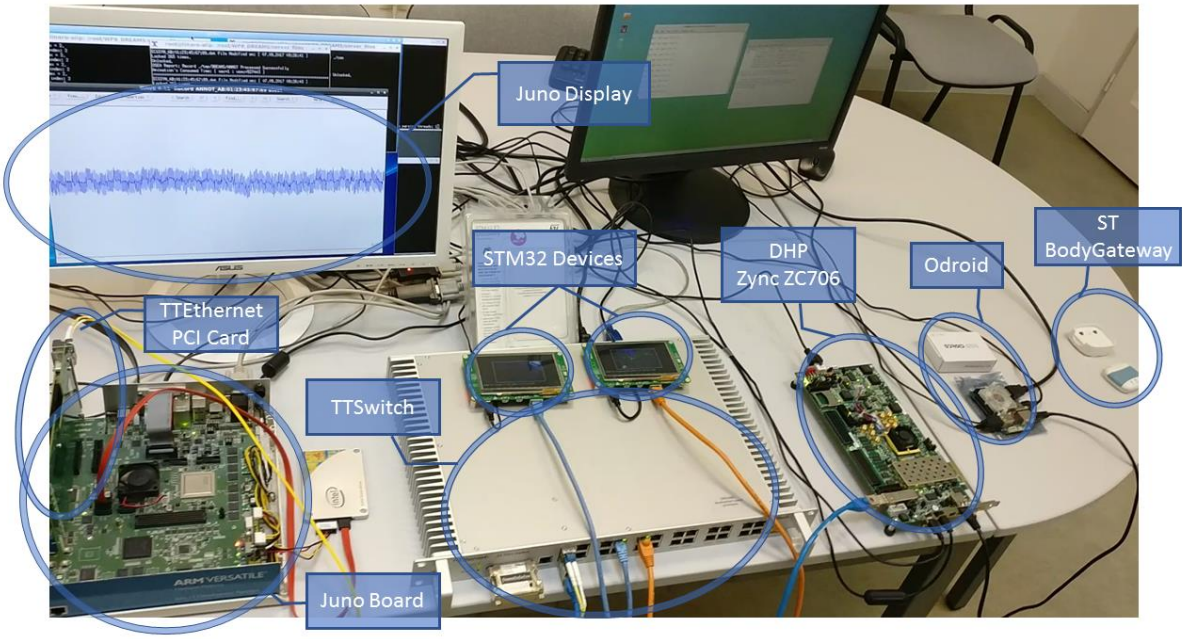


**Figure 2: Picture of the demonstrator**

The system for screening, prevention and management of disease is the system considered as critical in the Healthcare demonstrator. In addition, the use case involves also the streaming of video content to several rooms or patients. Thus, multiple services are supported by the Hospital Server:

- To enable content consumption by TV sets (using a wired network)
- To enable the ECG remote monitoring by receiving BGW information
- To have a single shared network supporting mixed critical traffic

Two main requirements must be addressed by the Hospital server architecture. The first one implies to guarantee real-time operations of the overall distributed system, which is composed by several subsystems with different criticality levels that are sharing the network and computational resources. Indeed, the Hospital server architecture should provide guarantees to meet the real-time constraints of critical information related to electrocardiography (ECG), while sharing the network with video streaming contents. Last but not least is the reliability requirement in presence of faults. Indeed, the Hospital server architecture should provide spatial and temporal isolation for the critical ECG application along with entertainment systems in order to ensure that the safety critical application is not affected by the non-critical one even in case of failure.

### 2.2.1 ARM Juno development platform

In this context, the ARM Juno Development platform combined with a TTEthernet device is used as a Hospital server. This platform offers significant flexibility in terms of development and prototyping features, while providing high computing performance and a reliable communication.

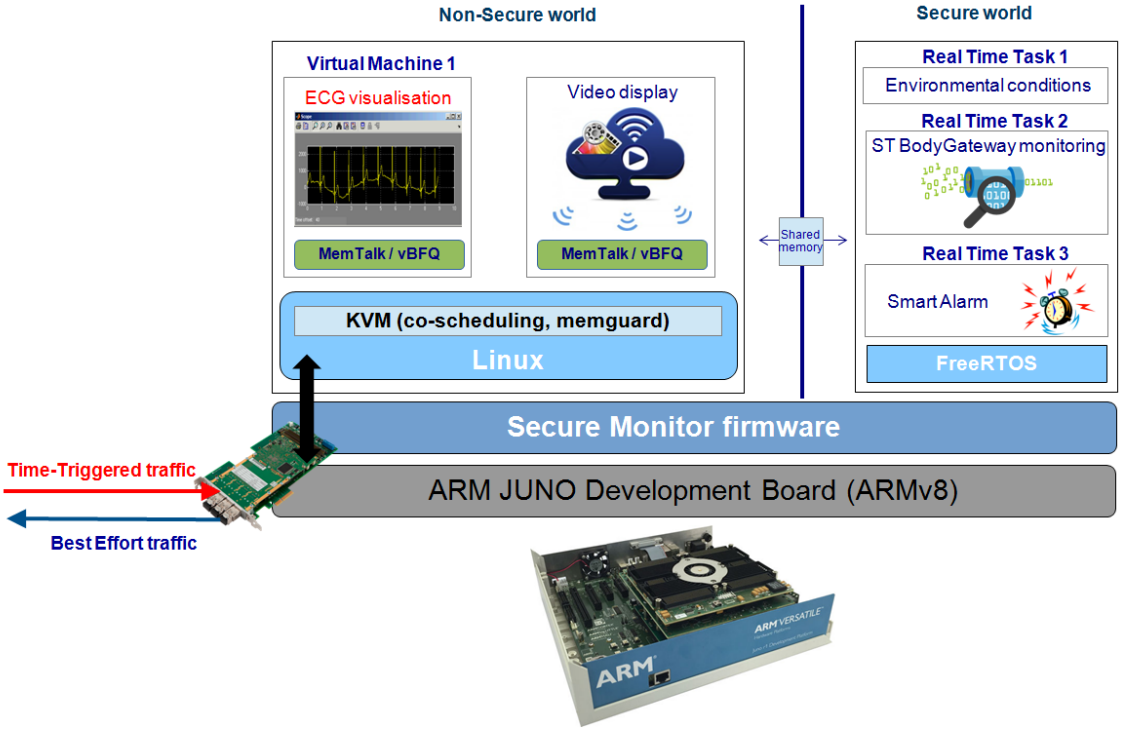


**Figure 3: Software architecture on the Juno board (Hospital server)**

Figure 3 shows that the Hospital server is composed by several DREAMS software technologies, which enable a safe architecture to consolidate ECG monitoring system along with video streaming applications.

To achieve flexibility, the TTEthernet device has been assigned to the Non-Secure world since all the necessary drivers to use this peripheral are already available for Linux. However, it is important to notice that the TTEthernet device should be handled by the Secure partition for a final product direction in order to ensure the processing of ECG data by the Real-Time Operating System (RTOS) even in case of a Non-Secure partition failure. However, such an implementation will require to port the drivers in the RTOS which requires too much effort according to the project timeframe.

Finally, this implementation enables continuous or intermittent physiological monitoring and detection of abnormalities arising from a range of medical conditions coming from different patients. As a matter of fact, the information sent by the BGWs can be examined by the doctor for the necessary intervention. Patients' data are logged via a dashboard and warnings are sent in real-time. The ECG visualization is executed in the execution environment of the Hospital server where other less critical applications, such as video streaming services, are concurrently executed.

### 2.2.2 DHP platform

The DREAMS harmonized platform (DHP for short) is a multi-core processor architecture developed by the DREAMS project. A prototype of the DHP using the Xilinx Zynq ZC706 FPGA Development board is used in the Healthcare demonstrator. Figure 4 shows the Zynq ZC706 board (on the left), where the connectors to access the Ethernet and TT-Ethernet (TTE) networks are highlighted. Additionally, the figure shows on the right a high-level view embedded within the prototype. In the Healthcare demonstrator only the TTE Controller and the dual-core ARM Cortex A9 tiles were required. Detailed information of the platform can be found in the *D1.2.1 Architectural Style of DREAMS*.

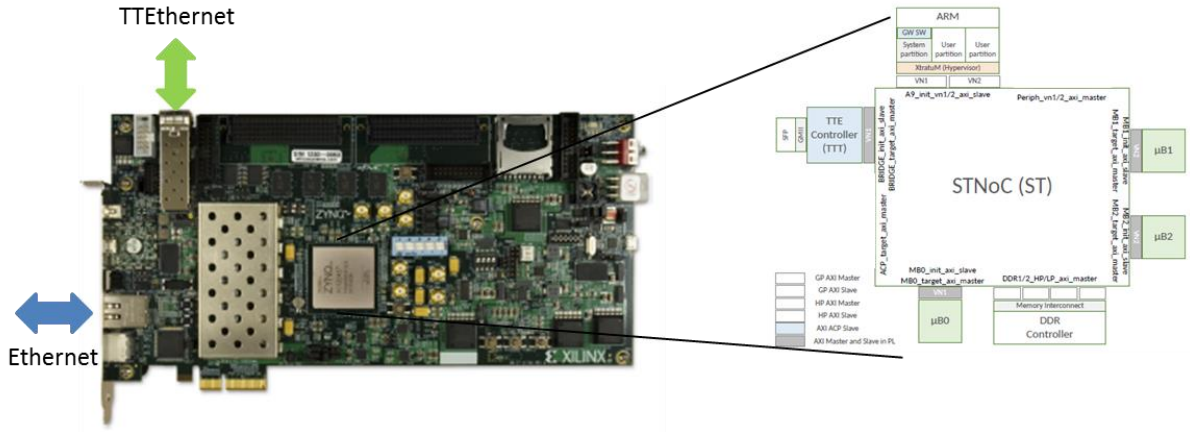


**Figure 4: DREAMS harmonized platform**

The XtratuM hypervisor is used in the DHP dual-core ARM Cortex A9 tile. Extended information can be found on *D2.3.1 XtratuM support of enhanced hypervisor layer services* and *D2.3.4 Hypervisor adaptation and drivers for local resource manager*. A multicore partition is used to manage the Ethernet and TTE traffic. The Ethernet software driver embedded in the partition uses the lwIP (lightweight IP) implementation, which is an open source TCP/IP stack designed for embedded systems and typically used on applications with real-time restrictions. The TTE software driver is embedded in the XtratuM hypervisor and it can be used by the partition in a transparent way, such as it is done for inter-partition communication (IPC) through queuing or sampling communication ports (see *D2.3.4*).

In the DHP, the Ethernet traffic received is routed to the TTE network and vice versa, the traffic received from TTE is routed to the Ethernet network. A TCP server is implemented in the XtratuM partition in order to establish a communication channel for the Ethernet traffic from Odroid board. Partition application is connected to a server provided by the Odroid board to create a route for the TTE traffic received from Juno board. A static table for the routing is defined in the partition application, where IP addresses/TCP ports are linked to XM ports/TTE virtual links.

The binding between TTE virtual links and XtratuM queuing ports is statically defined during an off-line design.

### 2.2.3 TTEthernet Network

The scheduling plan for the TTE network is based on the network traffic and the temporal requirements needed in the system. The section 3.1 defines a set of KPIs, which describes the type of traffic and the temporal restrictions of the components in the Healthcare demonstrator. The traffic in the ST Body Gateway is periodic and the transmission path is through Odroid, DHP and TTE switch until reaching the Juno board. Since a Bluetooth protocol stack is not available for the DHP an Odroid board has been introduced to address this requirement. The STM32 board uses the TTE switch to communicate with the Juno Board. Juno board sends control messages to the Odroid and STM board.

The path 1 in the Figure 5 requires only TT (Time-Triggered) slots for time-triggered traffic (see KPIs #22 until #31) between DHP and TTEthernet switch. It is important to point out that path 1 is one of the critical paths since it has to support several time-triggered flows. This requirement has been achieved using several TT slots. In the Path 2, the traffic required between Juno and STM32 discovery is best effort traffic or in other words, regular Ethernet traffic. In the path 3, the TT traffic and best effort traffic are mixed.

The temporal restrictions and TTE restrictions lead to define the following features:

- Several TT slots from DHP to Juno, each one associated to a Body Gateway with a Period of 10 ms
- A TT slot for traffic from Juno to DHP having a Period = 10ms

- Maximum message size => Payload = 1446 bytes
- A buffer is defined for each TT slot => 10 messages
- Based on previous characteristics (period and maximum message size), the maximum transmission bandwidth through the TTE network in the path 1 is *144600 bytes/s*, which is around *141 Kbytes/s*.

The temporal and functional restrictions for the path 2 are the same as for regular Ethernet.
In the path 3, the TT slots are defined and the regular traffic is sent together to the TT traffic. However, the delivery of the TT traffic is guaranteed and prioritized against the regular traffic.

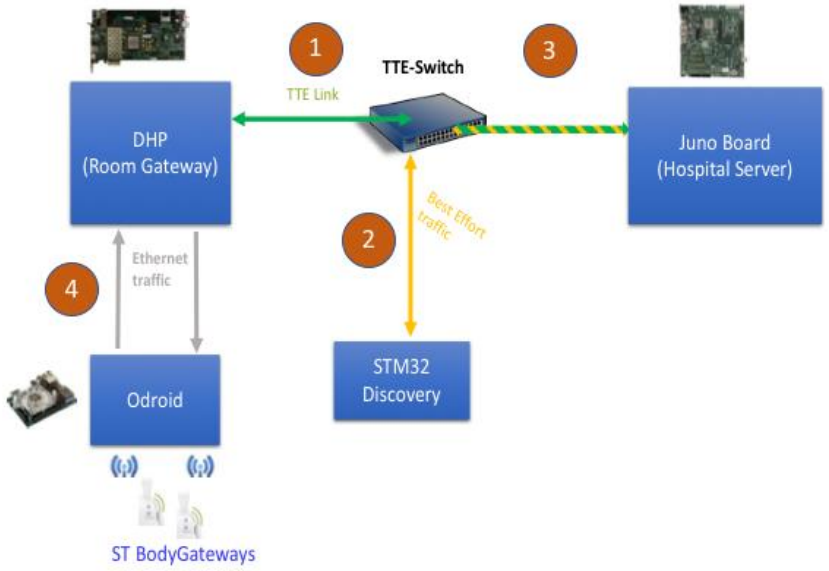


**Figure 5: Healthcare demonstrator highlighting the network paths.**

### 2.2.4 Ethernet network

This network is referred to the path 4, that is, to the traffic from Odroid board to DHP. This traffic is related to the number of Body gateways associated to each DHP. In general, we assume that a maximum of 6 devices may be associated to each DHP, which corresponds to having a maximum of 6 patients per room. Therefore, the scalability is guaranteed by different DHPs installed one per room. Periodic messages or a periodic and limited burst transmission is expected in this path.
The introduction of Odroid board has several complications in the management of traffic within the path 4 due to the intrinsic characteristics of the Ethernet network, where the medium access control is oriented to events and in the other extreme we have configured the medium access based on time. For this reason, it is important to take into account the maximum number of messages (i.e. the maximum bandwidth) configured in the TTE network in order to avoid message loss in the TTEthernet network, and hence, a bottleneck in the TTE transmission. The restrictions for this path can be extracted from the configuration information provided for the path 1 (section 2.2.3) and it can be summarized as:

- Maximum 10 messages (size of the TTE buffer) of 1446 bytes (maximum payload message size) can be transmitted at once and it requires short periods between transmissions based on the period of the TT slots (10ms).
- Burst transmission requires a high-bandwidth transmission over a short period. The formula to obtain the minimum time interval between burst transmission for the traffic from Odroid to DHP is:

$$Burst_{Interval} \geq (N_{msg} + 1) * Period_{TTSlot}$$

Where $\boldsymbol{N_{msg}}$ is the number of messages to be sent at once. $\boldsymbol{N_{msg}}$ must be less than the buffer size of the TT slot (10 messages in our scenario).
$\boldsymbol{Period_{TTSlot}}$ is the period of the TT slot (10ms in our scenario).

- Effective Bandwidth when the transmission of messages is not periodic but it is based on minimum time interval between burst transmission:

$$Effective\ Bandwidth = \frac{1446 * N_{msg}}{Period_{TTSlot} * (N_{msg} + 1)}$$

Where $\boldsymbol{N_{msg}}$ is the number of messages to be sent at once. $\boldsymbol{N_{msg}}$ must be less than the buffer size of the TT slot (10 messages in our scenario).
$\boldsymbol{Period_{TTSlot}}$ is the period of the TT slot (10ms in our scenario).

The issue with the burst transmission can be explained through an example. We are going to suppose that 5 messages require to be sent at once, so it requires to calculate the minimum time interval between burst transmission and the effective bandwidth:

$$Burst_{Interval} = (N_{msg} + 1) * Period_{TTSlot} = (5 + 1) * 10ms = 60ms$$

$$Effective\ Bandwidth = \frac{1446 * N_{msg}}{Period_{TTSlot} * (N_{msg} + 1)} = \frac{1446 * 5}{0.01s * 614} = 120500\ bytes/second$$

Figure 6 shows the flow of messages between Ethernet and TTEthernet for the example described above. Note in the figure that the time interval between burst transmission can have some +/- drifts (time 120ms and 240ms) but it does not have an impact in the TTE transmission. These drifts are considered in the calculation because the Odroid board uses the regular Linux kernel (without RT extensions), so the temporal requirements cannot be strictly fulfilled.

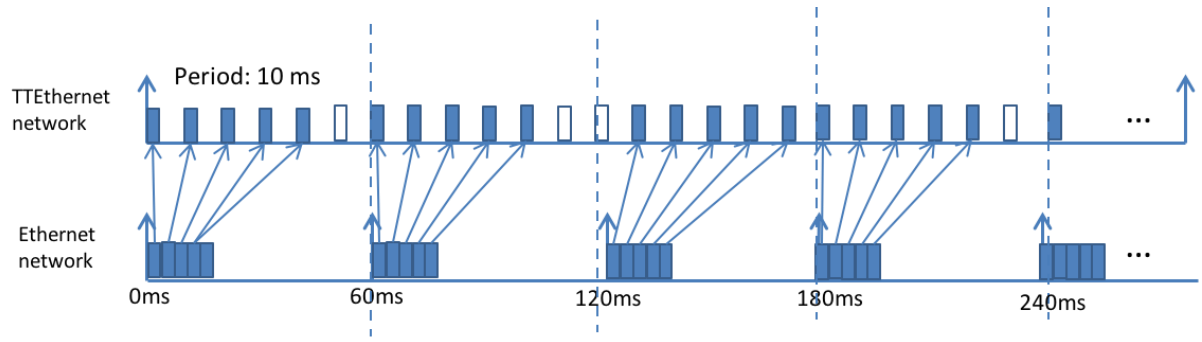


**Figure 6: Example of traffic from Odroid to DHP board based on burst transmission with a $\boldsymbol{Burst_{Period} = 60ms}$ and 5 messages per burst transmission.**

If the requirements listed previously are fulfilled, the traffic between Ethernet and TTE can be guaranteed.

### 2.2.5 Practical case and theoretical analysis

The defined number of devices supported per DHP is less than the maximum theoretical number of devices supported per DHP. The theoretical number can be calculated using the temporal and functional requirements of the ST BodyGateway in the transmission of the ECG raw data:

- The size per sample is 10 bytes
- 256 samples per second are generated by device, i.e. each device generates 2560 bytes per second.

Performing a theoretical analysis and assuming the worst case:

- 1 device requires 2560 bytes which means 2 messages (maximum payload message size equals to 1446)

If the way to send the information is through burst transmission in minimum time intervals, the maximum number of devices supported will depend on the number of messages sent on each burst transmission. Supposing different scenario and based on the formulas defined in the previous section:

1. The ECG raw information is sent at once for each device (2msg x 1 devices = 2msg):
   - $Burst_{Interval}$ = ( 2msg + 1) * 10ms => 2 messages each 30 ms
   - $Effective\ Bandwidth$ = (1446*2msg)/(0.01*(2+1))= 96400 bytes/s
   - With this burst transmission rate, the maximum number of devices supported would be: (#Devices on each burst)/$Burst_{Interval}$ = 1 device / 0,03 = **~33 devices**
2. The ECG raw information is sent at once for each 3 devices (2msg x 3 devices = 6msg):
   - $\mathrm{Burst}_{\mathrm{Interval}}$= ( 6msg + 1) * 10ms => 6 messages each 70 ms
   - $Effective\ Bandwidth$= (1446*6msg)/(0.01*(6+1))= 123942bytes/second
   - With this burst transmission rate, the maximum number of devices supported would be: (#Devices on each burst)/$Burst_{Interval}$ = 3 devices / 0,07 = ~**42 devices**

Therefore, the configuration proposed for the current scenario could handle more devices when the restrictions mentioned above are fulfilled.

Additionally, the period of the TT slot could be reduced if a higher bandwidth is required.

## 2.3 Experimental Framework on Zedboard (no DHP or TTEthernet)

In the in-hospital ECG processing use-case, doctors connect to a server for accessing ECG data of their patients.

In this subsection, we focus on a more limited home media gateway demonstrator (without DHP or TTEthernet) and evaluate WP2 technologies involving memory and network bandwidth regulation algorithms (called Extended MemGuard and NetGuard). The proposed algorithms implemented as GNU/Linux kernel modules (in x86 and ARM v7/v8) differentiate rate-constrained from best effort traffic and provide a mechanism for initializing (before the first period) and dynamically adapting (at periodic intervals) the guaranteed memory bandwidth per core or network bandwidth per connected (incoming or outgoing) network IP. The proposed strategies enhance support of mixed criticality applications on distributed embedded architectures by extending the current state-of-the-art in access control policies (genuine MemGuard algorithm), providing a guaranteed violation free operating mode for rate-constrained traffic, and supporting dynamic adaptivity through EWMA (Exponentially Weighted Moving Average) prediction. By examining a mixed-criticality scenario with real-time ECG processing and best effort video traffic on a hospital media gateway (Zedboard with two ARM Cortex-A9 cores), we show that simultaneous use of our MemGuard and NetGuard implementations enables fine-grain regulation of network and memory bandwidth for improved quality-of-service characteristics.

Next, Section 2.3.1 details the MemGuardXt/NetGuardXt extensions, including the algorithm, methodology and implementation. Section 2.3.2 details our healthcare use case running on the home media gateway. Section 2.3.3 summarizes our results and provides a summary and future extensions.

### 2.3.1 Linux Regulation Strategies

Network and memory bandwidth management strategies can improve performance of communication-intensive memory-bound computations in distributed embedded systems based on Multi-Processor Systems-on-Chip (MPSoCs) by contracting available resources and allowing bandwidth reclaim mechanisms to efficiently utilize unused bandwidth. These mechanisms when combined together with CPU bandwidth scheduling, a functionality already provided by the Linux kernel, provide the capability to apply holistic techniques to system resource management.

More specifically, network bandwidth regulation techniques allow differentiated services for communication-intensive applications through monitoring and control of packet communications. Traffic shaping, smart scheduling, congestion avoidance via admission control, reservation protocols and classification schemes can be used to avoid filling the network capacity. Similarly, memory bandwidth management schemes allow cores to share the memory hierarchy, avoid saturation or monopoly phenomena, and run memory-intensive programs more efficiently.

While previous approaches rely on specialized hardware subsystems to successfully manage shared resources, e.g. at memory [18][19][20][21] and network interface level [22], we concentrate on bandwidth regulation in Linux, without the design of additional hardware components. In critical hard real-time operating systems (e.g. in transportation or medicine) it is obligatory for certification reasons to completely avoid interferences, while in less critical systems running Linux it is often enough to ensure that such disruptions are not harmful. Thus, regulation policies aim at managing interference so that higher critical application tasks in a mixed-criticality environment will effectively fulfill a sufficient, predefined performance.

Within this context, MemGuard [23][24] performs dynamic memory bandwidth management at CPU-level by using hardware performance counters to monitor periodically the number of last-level cache misses (or equivalently accesses to the shared bus). In this deliverable, we introduce an extension to MemGuard algorithm (called MemGuardXt) which a) provides as an option a hard guarantee on the traffic rate which is especially important for real-time applications and b) improve its adaptivity for predicting bandwidth. We also improve modularity, by allowing our MemGuardXt algorithm to be used directly in either user- or kernel-space, in one or more instances. Using this methodology, we define two kernel modules: a) a kernel module running our MemGuardXt algorithm and b) a new network regulation module (called NetGuardXt) running over netfilter which uses a similar algorithm to MemGuardXt.

We have evaluated our modules in an actual mixed-criticality use case involving a) a distributed soft real-time ECG processing application that we have developed by extending the open source WFDB, OSEA and WAVE packages from PhysioNet, and b) incoming best effort video traffic on a hospital media gateway (Zedboard with two ARM Cortex-A9 cores). By focusing on both system and application metrics, such as NetGuardXt/MemGuardXt characteristics and real-time performance of ECG application, we show how simultaneous fine-grain control of network and memory bandwidth can result to improved quality-of-service characteristics that can help soft real-time ECG processing.

#### *2.3.1.1 Genuine MemGuard Principles and Related Extensions*

Genuine Memguard allows sharing guaranteed bandwidth over several cores using a dynamic reclaim mechanism. Using this mechanism, at the beginning of each period (`period`) cores are allocated part (or all) of their assigned bandwidth (according to history-based prediction) and donate the rest of their initially assigned bandwidth to a global repository (called `G`). Then, during the period, a core may obtain additional budget from `G` based on past traffic demand (history) and residual guaranteed bandwidth. This self-adaptive reclaim mechanism avoids over-provisioning, improves

resource utilization and is similar to extended self-adaptive Dynamic Weighted Round-Robin (DWRR) [25].

Since the guaranteed memory bandwidth within a period under worst-case conditions (`r_min`) is significantly less than the maximum attainable memory bandwidth (e.g., usually close to 20%), the algorithm also allows best effort traffic (BE), i.e., traffic in excess of `r_min`. Thus, once all bandwidth has been exhausted within a period, MemGuard supports two approaches to generate BE bandwidth which refers to bandwidth used after all cores (`i` in total) have utilized all their assigned budgets, before the next period begins. First, it allows all cores to freely compete for guaranteed bandwidth, by posing regulation until the end of the period. Second, it applies sharing of BE bandwidth proportionally to reservations. There is no explicit provision for best effort traffic sources in MemGuard algorithm. As long as `r_min` is not exhausted, genuine MemGuard allows sources with a zero reservation (or sources that have otherwise exceeded their reservation), to repeatedly extract guaranteed bandwidth from `G`, up to the configurable minimum allocation (`Q_min`).

#### *2.3.1.1.1 Genuine vs Extended MemGuard (MemGuardXt)*

The genuine MemGuard algorithm [23][24] targets average instead of peak bandwidth reservation which limits its use in real-time applications. More specifically, a rate-constrained (RC) flow may steal guaranteed bandwidth from other RC flows and even exhaust the global repository, while other RC-flows have not yet demanded their full reservation potentially leading to guarantee violations. Although genuine MemGuard supports a reservation-only (RO) mode that removes prediction and reclaiming and allocates to RC traffic sources their full reservation in each regulation period, this mode performs poorly in terms of resource allocation.

The proposed Extended MemGuard provides a hard guarantee option on the traffic rate which is important for real-time applications. This extension (called Violation Free mode or VF) restricts reclaiming budget from the global repository via function `overflow_interrupt_handler` if, as a result, it can cause guarantee violation for an RC-flow within the same period for one or more cores. Moreover, it considers RC, as well as BE criticality types of cores, although this feature is not examined in this work.

Finally, notice that genuine MemGuard supports limited adaptivity for predicting memory bandwidth requirements (few periods). Extended MemGuard supports a general EWMA scheme, which computes a weighted average of all past periods based on parameter $(\lambda)$ which determines the impact of history. EWMA prediction is pre-calculated for each core when a new period starts using the formula:

$z_t = \lambda * x_t + (1-\lambda) * z_{t-1}$, where $t>1$, $0 \leq \lambda \leq 1$ and $z_1 = x_1$,

where $Z_t$ is the predicted bandwidth for the next period $(t+1)$, while $x_t$ is the consumed bandwidth from the core at the end of the current period (t). This formula better adapts to intermediate traffic perturbations, i.e. between short bandwidth fluctuations and abrupt changes.

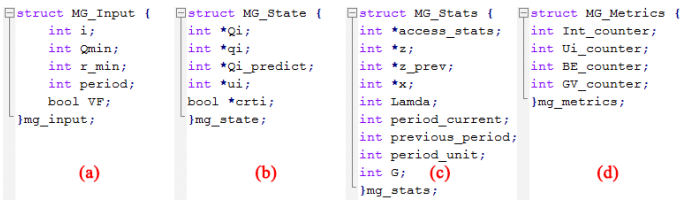


**Figure 7: MemGuard input parameters, system state, statistics and metrics.**

We next explain the rationale in our MemGuardXt and NetGuardXt implementation as Linux kernel modules on x86_64 and ARM platforms (32 and 64-bit). Unlike genuine MemGuard, our MemGuardXt/NetGuardXt implementation is modular and different instances can be easily used from either user- or kernel-space. Its core functionality is implemented as a separate, self-contained package (.c and .h files) implemented in ANSI C. For example, as shown in Figure 7, MemGuardXt supports four data structures:

- MG_INPUT data structure with input parameters `i`, `Q_min`, `r_min`, `period`, `and VF`; this info can be dynamically modified via `debugfs` and update action is taken when the next period starts.
- MG_STATE with initial, current, predicted and total used bandwidth and a criticality flag (set to true for RC traffic and false otherwise) `Qi[]`, `qi[]`, `Qi_predict[]`, `ui[]`, and `rc_flag[]`. If left uninitialized, `Qi` for all cores automatically takes the value of `r_min/i`. MemGuard algorithm distinguishes between RC and BE cores using the `rc_flag[]` array which denotes the criticality level of each core. Notice that both RC and BE cores can consume guaranteed and generate best effort traffic.
- MG_STATS related to EWMA prediction algorithm with $z_t$, $z_{t-1}$, $x_t$, $\lambda$, `previous_period (t-1)`, `current_period (t)`, `period_unit`, and `G`; Notice that, if called from kernel mode, EWMA bandwidth prediction is implemented using integer numbers only (instead of double) for optimization.
- MG_METRICS with number of interrupts, i.e. when there is a request for reclaim, used bandwidth (from all cores), best effort bandwidth and the number of guarantee violations, when guaranteed bandwidth has been donated and already consumed by others; notice that if `VF` is set, then `GV=0`.

The above data structures and functions are used from a Linux kernel module which provides the necessary monitors and actuators: e.g. timers, cache metrics and throttle mechanisms for MemGuardXt, or timers, bandwidth metrics and accept/drop functionality for NetGuardXt. Basic operation of MemGuardXt kernel module (left) with its core algorithm (right) is shown in Figure 8.

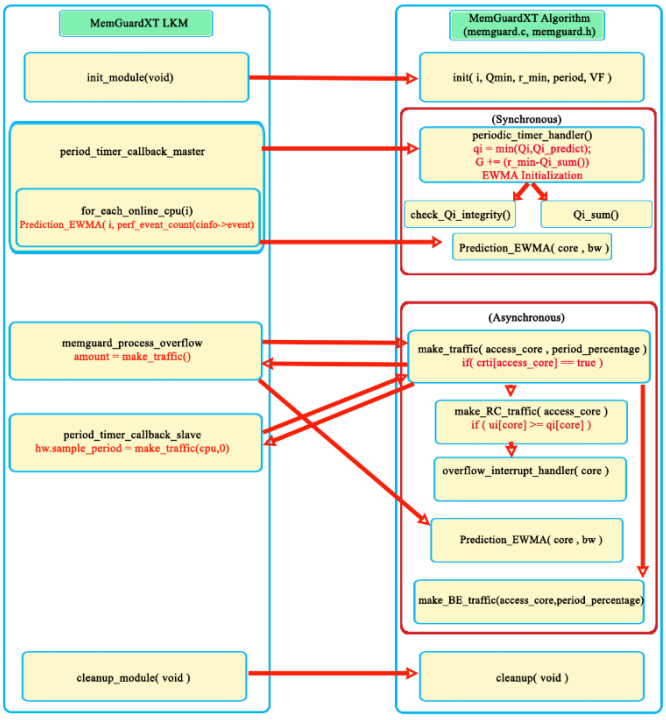


**Figure 8: MemGuardXt Linux kernel module (LKM) and core algorithm**

During MemGuardXt module insertion and removal (insmod/rmmod), `init_module` and `cleanup_module` functions invoked in the kernel driver also call corresponding functions in the core for initialization and memory cleanup. Periodically, `Prediction_EWMA` function is called to update the bandwidth consumed by each core based on the previous period and `periodic_timer_handler` resets all necessary statistical variables and reassigns the estimated bandwidth per core. This information is extracted from MemGuardXt algorithm by calling `make_traffic` from `period_timer_callback_slave` when the period starts. This value is increased on the fly by asynchronous calls of `make_traffic` from

`memguard_process_overflow` of the LKM module which also informs `Prediction_EWMA` that previously assigned bandwidth is already consumed.

#### *2.3.1.1.2 NetGuard Extension (NetGuardXt)*

We have also developed a different incarnation of Extended MemGuard as a Linux kernel module that uses custom netfilter hooks, the packet filtering framework built around `sk_buff` in Linux kernel. This allows independent kernel-level monitoring and control of network bandwidth of incoming and outgoing network flows using two separate NetGuardXt algorithm instances. Each such instance may define its own source/destination client IPs and bandwidth rate (`r_min`, `Qi`). This kernel module (called NetGuardXt) supports the existing API of Extended Memguard to provide network bandwidth regulation on Linux on x86_64 and ARMv7; our implementation on ARMv8 (64-bit Dragonboard 410c) has failed due to currently limited Linux kernel 4.0+ support of iptables/netfilter. While currently the period, number of traffic sources per interface and EWMA parameters can be set directly from the module as needed, other parameters (`r_min`, `Q_min`, and `Qi`) can also be configured on the fly, separately for each flow direction (outgoing and incoming) using debugfs. For example, the command

***echo "7000 500 3000 4000 15000 500 5000 10000" > /sys/kernel/debug/neguard/netguard_config***

configures NetGuardXt outgoing traffic to {`r_min`, `Q_min`, $Q_0$, $Q_1$} = {7000, 500, 3000, 4000} bytes/period (and similarly for incoming traffic).

NetGuardXt provides statistics concerning instant and cumulative statistics of accepted and dropped traffic (in packets or bytes per flow direction and connected client).

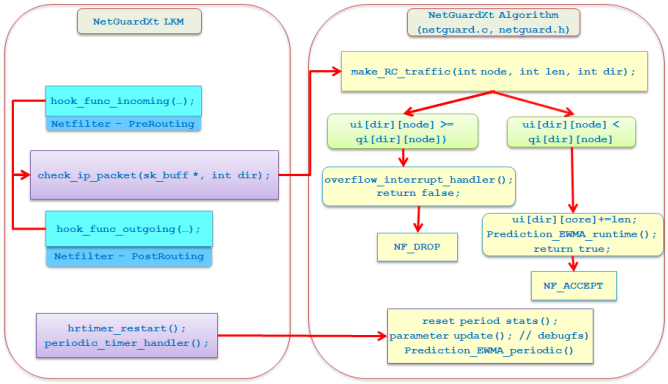


**Figure 9: NetGuardXt Linux kernel module (LKM) and core algorithm.**

Without delving into Linux kernel details (which involves understanding network drivers, netfilter for packet filtering hooks, high resolution timers, debugfs etc), we describe the main concepts of NetGuardXt and provide its API in Figure 9.

Each packet destined to a network client (incoming or outgoing) can be counted and checked using `bool make_rc_traffic` function. Packet is sent (`NF_ACCEPT`) if this function returns `TRUE`. Otherwise, the packet is dropped (`NF_DROP`). Counters are reset at the end of each period (function `period_timer_handler()`). A high-resolution timer (`hrtimer`) implements the period.

Similar to Extended MemGuard interface, the EWMA update functions `Prediction_EWMA_periodic` and `Prediction_EWMA_runtime` are used to adjust the predicted bandwidth per client. In cases where the requested bandwidth exceeds the given "budget" `overflow_interrupt_handler` is called to reclaim unused bandwidth from the global repository where donations may occur.

### 2.3.2 WP8 Use-Case: Hospital Media Gateway

In order to demonstrate use of our kernel modules, we consider a realistic healthcare use case composed of a mixture of *medical-critical tasks* related to ECG processing and *non-critical parts* related to patient entertainment. This use case combines different types of devices sharing the same network and processing infrastructure and provides a viable solution that evolves the traditional hospital entertainment system, called linear TV, towards an on-demand system.

### *2.3.2.1 Infotainment Functionality*

A Hospital Media Gateway (HMG) solution must also address end user needs for infotainment evolving the existing hospital entertainment system (called linear TV) located in each room to an on-demand distributed system. Thus, in addition to medical-critical services involving healthcare data acquisition, analysis, privacy protection and continuous physiological monitoring of the overall health and well-being via STMicroelectronics BodyGateway device (BGW), the HMG must also involve as a core function transmission of non-critical premium content to HMG and eventual consumption by patients located in different rooms. With this aspect, a number of smart devices can act as video clients (using a wired network) with regard to content services eliminating the need for a dedicated high-end set-top-box per each end-device.

### *2.3.2.2 Single and Multi-Room Scenario (in-Hospital)*

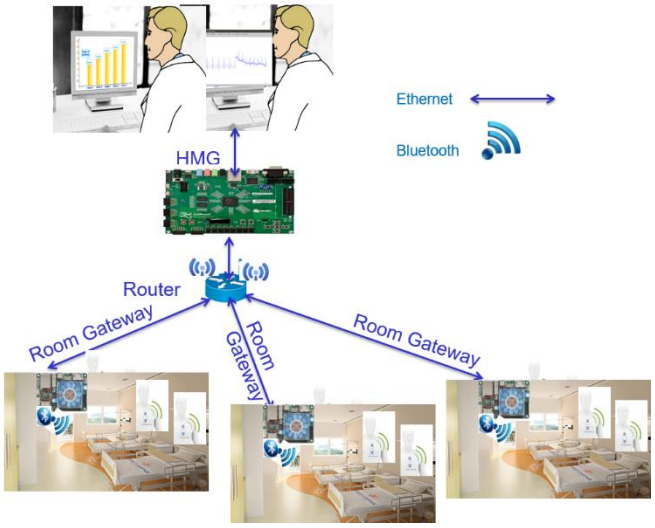


**Figure 10: A Hospital Media Gateway (HMG) in a multi-room scenario**

In the single room scenario, medical patient data from up to 8 patients collocated in the same room who wear the BGW is transmitted via Bluetooth to local room gateways (32-bit Odroid XU4 board). Patients may move around hospital rooms and corridors as long as the Bluetooth signal-to-noise ratio is acceptable (usually 100m range). Figure 10 demonstrates the platform architecture that maps each local room gateway to a 32-bit Odroid XU4 board.

The single-room distributed network architecture can be extended to a multiple room scenario as follows. First, a wireless Bluetooth connection is used to transmit sensor data from multiple BGW gateways corresponding to patients in each room to a room gateway. Then, this data is transmitted from each room gateway (32-bit Odroid XU4) to the HMG (32-bit Zedboard and eventually 64-bit ARM Juno board) via the hospital's wired distribution network.

In the above in-hospital use case, we must guarantee soft real-time operation of the overall distributed system, consisting of subsystems (healthcare architecture and multimedia) with different criticality levels that are sharing network and computational resources. Criticalities extend from computation components (Media Server, BGW, ECG processing) to the communication infrastructure (wired and wireless network). Another important requirement is scalability of the healthcare architecture without affecting system properties, e.g. real-time requirement. Notice that the number of BGW devices (resp. software components) can scale depending on the number of patients to be monitored while the system is running.

#### *2.3.2.3 Real Time ECG Analysis and Visualization*

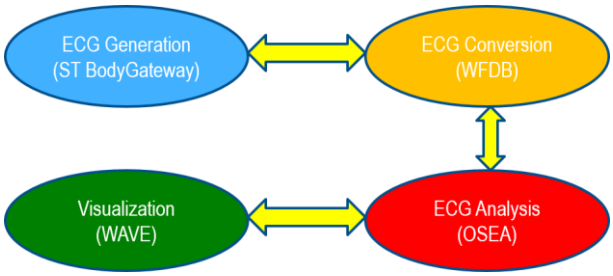


**Figure 11: The ECG analysis process.**

Our soft real-time ECG analysis system is based on open source software, in particular a medical decision support subsystem which provides annotation on the basis of non-fatal arrhythmias. Detection and classification of heart beat signal is the objective of the algorithms designed to support full medical decision support systems by diagnosing health issues and providing information to physicians. Thus, the system is also able to identify alarming situations by annotating critical situations graphically along with the ECG signal.

In our use case, an ECG analysis process is initiated on the Zedboard upon acquisition by a server component of the heart beat signal transmitted by the Odroid XU4 board. This signal is initially received by the Odroid board via Bluetooth from the BGW sensors and appended to a file prior to transmission. As shown in Figure 11, ECG analysis proceeds to invoke our custom real-time extensions on WFDB and OSEA, two open source software libraries whose main task is (offline) normalization of the input signal according to standard EC-13, and heart beat detection and classification. Finally, visualization is performed using WAVE, a fast-open source application based on XWindows and XView open source client toolkit (see Figure 11). WAVE supports fast, high resolution display of ECG waveforms at different scales with asynchronous display of the annotations. It also handles remote access by Web browser, but this feature is not used in our tests.

More specifically, since the BGW transmits an ECG signal consisting of 256 samples per sec, we first perform conversion to a standard ECG-13 compliant format using WFDB's wrsamp function which outputs two files: a standardized ECG signal "synth.dat" and an ASCII "synth.hea" file which contains info about ECG data stored in the previous file e.g. the total samples.

Then, we use a custom real-time version of easytest algorithm (part of OSEA, a.k.a. Open Source ECG Analysis) to a) dynamically change the sampling rate to 200 samples/sec and b) perform on-the-fly standardized ECG13-compliant heart beat detection and classification using different types of filters (e.g. noise reduction, QRS, SQRS) and related computations (R-R interval). This process allows classification of the ECG signal to Normal or Ventricular in "`synth.atest`" file, cf. [26][27]. OSEA uses Harvard's WFDB Physionet software which provides stable, self-adaptive automated ECG detection and analysis with very high positive predictivity (tested with MIT/BIH and AHA arrhythmia databases).

### 2.3.3 Experimental Framework and Results

In this section, we examine the effect of simultaneous memory/network bandwidth regulation (using our MemGuardXt and NetGuardXt kernel modules) on the above realistic in-hospital use case based on a hospital media gateway (Zedboard with two ARMv7 cores).

More specifically, in our prototype use case, we apply NetguardXt to regulate two types of incoming traffic on the Home Media Gateway (Zedboard):

- video-on-demand traffic arriving to Zedboard from an external server via an Ethernet router; this video is eventually distributed to clients via video streaming; notice that video streaming does not generate significant memory bandwidth (1-2 MB), thus in this deliverable we focus (without loss of generality) on incoming traffic. However, the same prototype has been used to control quality-of-delivery of outbound video traffic, see Figure 12.
- ECG network traffic, originating from two BGW devices connected to an Odroid XU4 and arriving to Zedboard via the Ethernet router.


**Figure 12: Regulating video streaming quality using NetGuardXt.**

ECG processing at the Zedboard involves running a `server` that opens independent TCP connections to receive new ECG data from BGW devices via Odroid XU4, together with an initial consumer process that starts WAVE application for each connected client, and an animator process (`wrsamp`, `easytest`) that sends asynchronously (via wave-remote function) an annotated ECG signal to WAVE. Notice that all these processes run on CPU0, while video-on-demand process runs on CPU1. Since Zedboard has only two CPUs available, both cores are considered rate-constrained. In our experiments,

- MemGuardXt configuration uses a fixed `period=1 msec, i=2, λ=0.2, r_min=`$Q_0$`+`$Q_1$`=90M/sec, Q_min=50`, while the MemGuardXt $Q_0/Q_1$ ratio (for cores 0 and 1) is set by one of three scripts; furthermore, since Zedboard does not provide a counter for last level cache (L2) cache misses, we have disabled L2 cache.
- Similarly, NetGuardXt configuration uses a fixed `period=1 sec, i=2, λ=0.2, r_min=`$Q_0$`+`$Q_1$`=70KB/sec, Q_min=1000`, while the NetGuardXt $Q_0/Q_1$ ratio (for cores 0 and 1) is controlled dynamically by the same three scripts described below.
- The three scripts which are the driving force behind our experiment *first fix MemGuardXt* $Q_0/Q_1$ *ratio* as `25/65, 50/40`, or `75/15`, and *then periodically, every 20 sec, reconfigure the NetGuardXt* $Q_0/Q1$ *ratio* (for cores 0 and 1) always with the same sequence: `{18/72, 16/74, 14/76, 12/78, 10/80, 8/82}`. Thus, each script gradually decreases the assigned bandwidth rate for ECG (and increases that of Video), while keeping a fixed ratio for ECG to Video memory bandwidth. Each running script is named after the fixed MemGuardXt configuration value, i.e. **MG 25/65**, **MG 50/40**, or **MG 75/15**.

Selection of memory/network $Q_0/Q_1$ rates for MemGuardXt and NetGuardXt was based on initial experiments that considered performance of each component (ECG processing or video traffic) in isolation, to locate regions in the system parameter space where mixed criticality effects are interesting.

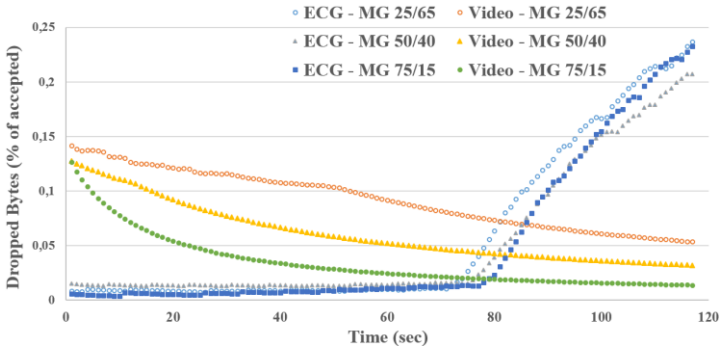


**Figure 13: Dropped Bytes for ECG and Video (as percent of accepted ones) while the corresponding script runs, i.e. ECG rate decreases every 20 sec.**

Figure 13, extracted from kernel logs, shows that gradually decreasing ECG network bandwidth via NetGuardXt from 18000kB/sec to 8KB/sec (in 20 sec intervals), results in an increasing cumulative ECG drop rate and decreasing drop rate of video traffic.

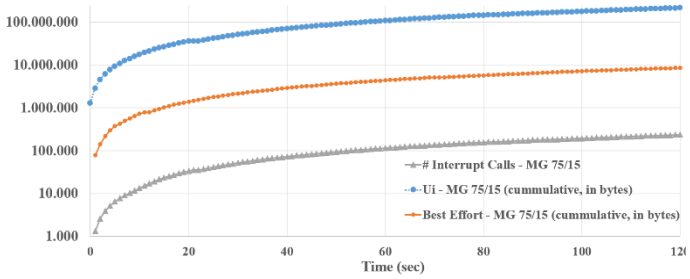


**Figure 14: Performance of MemGuardXt (MG 75/15 configuration); the vertical Axis units are in bytes for Used Bandwidth (Ui) and Best Effort.**

In Figure 14 we show corresponding MemGuard performance for MG75/15 case. Notice that in this case all MemGuard figures scale well despite the decreased bandwidth, i.e. TCP retransmissions due to drops at the incoming network interface (see Figure 13) appear to be still manageable in real-time.

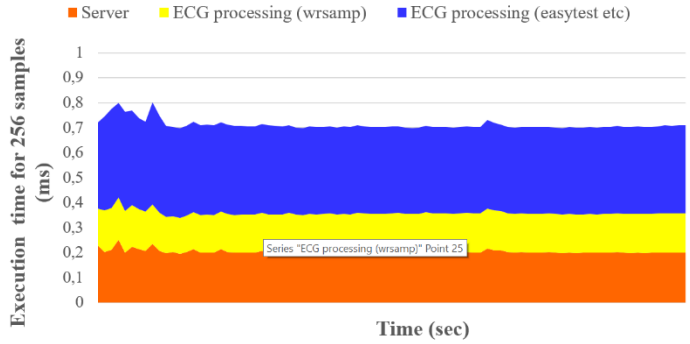


**Figure 15: Delays at Home Media Gateway (Zedboard) for MG75/15 script.**

Figure 15 shows an execution trace at the server assuming the same MemGuardXt configuration MG 75/15. Notice that we have set affinity so that ECG `server`, `consumer` and `animation` processes (involving `wrsamp` and `easytest`) all share ARM Cortex-A9 CPU0, while the video-on-demand service transferring files to Zedboard for further delivery runs on CPU1. Although ECG network rate is reduced, results are similar. They show that a proportionally larger share of the total execution time (up to 50%) is spent for `easytest` which performs filtering and asynchronous annotation of ECG data than either `server` which saves the ECG signal locally (approximately 30%), or `wrsamp` which performs signal conversions to EC-13 standard (20%). Small variations at the server can be attributed to acquiring file locks at the `server` and `animation` process (below 20ms with rare spikes).

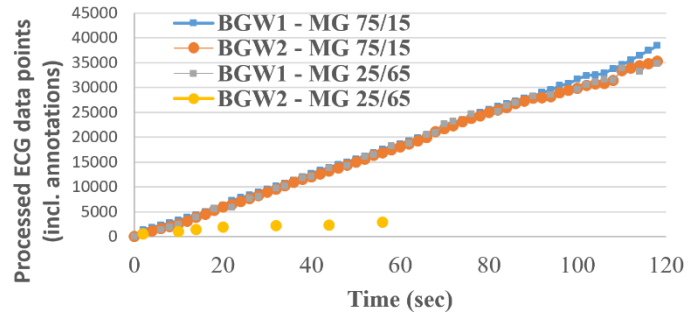


**Figure 16: Real-time performance of ECG processing application for two BodyGateway devices for two MemGuardXT configurations.**

Finally, Figure 16 compares the amount of ECG data delivery from each of the two BGW devices to the animator (WAVE application). It is clear that while for MG 25/65 configuration one of the BGW devices has completely stopped due to memory bandwidth starvation, in the MG75/15 configuration the animator is able to process traffic from both BGW devices in soft real time.

Overall, system-level bandwidth regulation algorithms can differentiate among rate-constrained and best effort traffic sources in systems-on-chip. Our work extends existing memory bandwidth regulation policies (MemGuard) by providing improved adaptivity through EWMA prediction and considering a violation free operating mode for rate-constrained flows. Our implementation follows a highly modular approach, allowing our MemGuard extensions (MemGuardXt) to be used directly in either user- or kernel-space, in multiple instances. By applying this engineering approach, we have also designed a network bandwidth regulation module (called NetGuardXt) running over netfilter which uses a similar algorithm to MemGuardXt to control incoming or outgoing traffic per IP.

We have considered combined MemGuardXt/NetGuardXt effects in a mixed-criticality use case involving a hospital media gateway prototype (Zedboard with two ARM Cortex-A9 cores). The media gateway simultaneously performs *soft real-time processing and annotation of ECG signals* from STMicroelectronics' BodyGateway pulse sensors (relying on our extensions on the open source WFDB, OSEA and WAVE framework from PhysioNet) while also *storing video-on-demand traffic for video streaming*. By examining different NetGuardXt and MemGuardXt configurations, we have shown how fine-grain control of network and memory bandwidth can help soft real-time ECG processing.

Our future plans include extending our use case to a more powerful hospital media gateway based on ARM Juno board that would enable use of rate-constrained and best-effort cores for both ECG and video streaming applications.

An alternative implementation in the heart of the Linux scheduler would allow MemGuard to regulate rate-constrained and best-effort traffic at process- instead of core-level. This new scheduling policy is currently being implemented on embedded ARMv7 technology (Zedboard running Linux kernel 3.17).

# 3 Evaluation Methodology

In this chapter, we focus on the methodology to evaluate the project approach on the basis of a Healthcare demonstrator integrating all hardware components (ST Bodygateway, Odroid, DHP, TTEthernet and Juno) as shown in Figure 1. An assessment is achieved in this report through the definition and the evaluation of Key Performance Indicators (KPIs) based on the general project objectives defined in Description of Work [5] Part B, section 1.1.

## 3.1 Key Performance Indicators (KPIs)

KPIs are regarded as a collection of metrics for quantifying the objectives of the project, monitoring its activity progress and assess the expected results.

The KPIs presented in this section are expected to be:

- Objective: it shall be possible to measure them objectively.
- Measurable: it shall be possible to quantify them.
- Relevant to the project: the partners shall confirm their interest.
- Comparable: to the situation of the application use case before using DREAMS approach and technologies.

The performance indicators defined in the following tables will be traced to one or more measures for success. In this preliminary evaluation, they will provide quantitative information to support the qualitative evaluation of every measure for success. Some of the measures for success are not traced to any KPI, since there may be no quantitative data that could support the conclusion.

The KPIs are classified into three subsets:

- **'D'**: The KPIs marked with 'D' can be evaluated in the preliminary and final reports (e.g., jitter, boot time, etc).
- **'E'**: These KPIs can only be evaluated in the final report at the end of the project (e.g., Percentage of DREAMS building blocks used by the demonstrator, etc)
- **'A'**: These KPIs can be objectively evaluated only after the project since some experience with the technology is needed (e.g., Time-to-market reduction of a mixed-criticality system based on DREAMS architecture and technologies). However, an estimation is provided in this report.

Table 1 lists and describes all KPIs of the project, and traces all of them to the measures for success they aim at providing arguments for evaluation. The last column indicates when this metric can be obtained:

| ID | KPI | Description | Measure for Success[1] | Time |
|---|---|---|---|---|
| 1 | Achievable Safety Integrity Level | Maximum achievable Safety Integrity Level (e.g. ASIL-B, ASIL-C) according to ISO 26262 [6][7][8] for the secure monitor firmware layer | 1.1, 2.7 6.1, 6.2 | D |
| 2 | Validated support for key real-time OS | (Boolean) The ARM JUNO development platform supports integration of FreeRTOS to be used as the OS for the supervision. | 1.2 | D |

[1] Detailed description of the measures for success can be found at Section 3.2.

| ID | KPI | Description | Measure for Success[1] | Time |
|---|---|---|---|---|
| 3 | Maximum jitter induced by the secure monitor layer | Bounded value for jitter in the execution of the most critical real-time thread | 1.2 | D |
| 4 | Maximum overhead during the RTOS boot | Bounded value for overhead induced by the secure monitor firmware layer during the boot of the RTOS | 1.2 | D |
| 5 | Temporal and spatial isolation by construction | (Boolean) The safety concept (supported by the verification plan) demonstrates that the architecture provides temporal and spatial isolation of partitions by construction | 2.1, 2.7, 3.1, 6.1 | D |
| 6 | Maximum latency overhead of applications inside a KVM virtual machine | Percentage of the overhead of the latency of KVM virtual machine on loaded system. Latency is measured with Linux tool “cyclictest” inside a virtual machine with and without CPU workload. The overhead is the difference between those two measurements. | 2.1, 2.4 | D |
| 7 | I/O latency inside KVM virtual machine is not affected by the I/O workload | (Boolean) The I/O latency of application inside virtual machine, on a system with I/O workloads, is about the same value than on a system with idle medium. | 2.1 | D |
| 8 | Memory bandwidth isolation by construction | (Boolean) The architecture provides a memory bandwidth isolation between tasks | 2.1 | D |
| 9 | Memory bandwidth reservation for highest criticality level application | (Boolean) The architecture provides a memory reservation feature to preserve memory bandwidth of highest critical applications | 3.1 | D |
| 10 | Fault containment by construction | (Boolean) The certification body accepts evidences to demonstrate fault containment by construction | 1.3,1.1 | E |
| 11 | Percentage of system architecture/design modelled | Percentage of the system architecture and design that is able to be modelled with the tools developed in DREAMS | 1.7 | E |
| 12 | Percentage of software application modelled | Percentage of the application software that is able to be modelled with the tools developed in DREAMS | 1.7 | E |
| 13 | Bounded temporal network routing. (TTEthernet -> Ethernet) | Delay introduced in the path of data packets when they are routed from the TT-Ethernet network to the Ethernet network through the DHP board. | 2.3 | E |
| 14 | Bounded temporal network routing. (Ethernet -> TTEthernet) | Delay introduced in the path of data packets when they are routed from the Ethernet network to the TT-Ethernet network through the DHP board. | 2.3 | E |
| 15 | Bounded temporal interference (network) | Delay introduced in the safety-related communications when heavy non-safety traffic (video) is generated in the network | 2.1 | E |

| ID | KPI | Description | Measure for Success[1] | Time |
|---|---|---|---|---|
| 16 | Bounded temporal interference (processing) | Delay introduced in the critical thread of the safety-related partition when heavy processing load is generated in neighbouring non-safety partitions | 2.1 | E |
| 17 | Bounded temporal interference (resources access rate) | Delay introduced in the access to resources (memory) by the safety-related partition when heavy resource consumption is required by neighbouring non-safety partitions | 2.1, 2.2 | E |
| 18 | ST Body gateway-to-partition latency | Latency between a value is read at the sensor and delivered at the partition where it is going to be processed | 2.5 | E |
| 19 | Percentage of development steps covered by tools in demonstrator | Percentage of development steps where DREAMS tools provide support in the demonstrator, in one or more of the following aspects: safety, timing, energy, variability | 4.2 | E |
| 20 | Percentage of automatically executable transformations | Percentage of automatically executed transformations between consecutive development steps provided by tools | 4.3 | E |
| 21 | Adaptability to evolution of product and standards | (Boolean) The approach provides required adaptability for evolution of product and standards | 5.6 | A |
| 22 | ST Bodygateway ECG raw data | Real-time constraint 128/256 Hz | 1.1 | E |
| 23 | ST Bodygateway Heart Rate | Real-time constraint 1 each 10/15/30/60 sec | 1.1 | A |
| 24 | ST Bodygateway Heart Rate Reliability | Real-time constraint 1 each 10/15/30/60 sec | 1.1 | A |
| 25 | ST Bodygateway R-R Variability | Real-time constraint 1 each 10/15/30/60 sec | 1.1 | A |
| 26 | ST Bodygateway BIOZ | Real-time constraint 32 Hz | 1.1 | A |
| 27 | ST Bodygateway ACC XYZ | Real-time constraint 50hz | 1.1 | A |
| 28 | ST Bodygateway Body Position | Real-time constraint 1 each 5/10/15/30/60 sec | 1.1 | A |
| 29 | ST Bodygateway Activity level | Real-time constraint 1 each 5/10/15/30/60 sec | 1.1 | A |
| 30 | ST Bodygateway Breathing Rate | Real-time constraint 1 each 15/30/60 sec | 1.1 | A |
| 31 | ST Bodygateway Battery | Real-time constraint 1 each 10/15/30/60 sec | 1.1 | A |

| ID | KPI | Description | Measure for Success[1] | Time |
|---|---|---|---|---|
| 32 | Juno R1 CPU utilization in video streaming – Maximum overhead | CPU utilization to achieve a required frame-rate quality on the STM32F746G-DISCO (2 scenarios) – AVI video rendering and streaming raw bitmap images (not jpeg) application pinned to A57<br>Real-time constraint for:<br>a) 24 FPS, half-screen size, 24-bits/pixel, peak=90%, avg=85% (A57 cluster)<br>b) 20 FPS, half-screen size, 16-bits/pixel | 1.4 | E |
| 33 | Juno R1 memory utilization in video streaming – Maximum overhead | Memory utilization to achieve a required frame-rate quality on the STM32F746G-DISCO (2 scenarios) – AVI video rendering and streaming raw bitmap images (not jpeg) application pinned to A57<br>Real-time constraint for:<br>24 FPS, half-screen size, 24-bits/pixel: 240MB | 1.4 | E |
| 34 | Juno R1 – STM32F746G-DISCO Ethernet network utilization in video streaming – Maximum overhead | Ethernet (UDP) network bandwidth to achieve a required frame-rate using raw video for half-screen size of STM32F746G-DISCO<br>Real-time constraint for:<br>24 FPS, half-screen size, 24-bits/pixel: 80Mbps | 1.4 | E |
| 35 | STM32F746G-DISCO CPU utilization in video streaming – Maximum overhead | CPU utilization to achieve a required frame-rate using raw video in STM32F746G-DISCO without JPEG accelerator, DMA to framebuffer.<br>Real-time constraint for:<br>24 FPS, half-screen size, 24-bits/pixel: 75% | 1.4 | E |
| 36 | Real-time characteristics of ECG Processing application | Related to ECG data analysis for automated cardiac disease detection and visualization, soft real-time operations of the overall distributed system must be guaranteed since the healthcare demonstrator includes subsystems with different criticality levels (healthcare data and multimedia). | 1.2 | E/A |
| 37 | Scalability of the healthcare architecture in terms of number of Body Gateway devices | Number of ST body gateway devices that can be simultaneously connected to the platform without affecting real-time constraints. | 3.5 | E/A |

**Table 1: Key Performance Indicators**

Table 2 collects the values of the KPIs, evaluated or estimated. According to the KPI type (i.e., Boolean or not), some results have been measured while others have been determined through the documentation. Additional information is provided in the comments column.

| ID | KPI | Goal | Value | Comments |
|---|---|---|---|---|
| 1 | Achievable Safety Integrity Level | ASIL-C | Under functional safety assessment | The secure monitor firmware layer has been designed to meet the stringent requirements of the ISO 26262 standard. The certification target of this software component is ASIL-C, which corresponds to SIL-2/SIL-3 of the Functional Safety Standard IEC61508 that fits the needs of the healthcare use-case. The Audit phase 1 related to the concept has been passed with success and the Audit phase 2 is ongoing. |
| 2 | Validated support for key real-time OS | Yes | Yes | The support of FreeRTOS, which is the monitoring real-time OS for Healthcare demonstrator, is fully validated on the ARM JUNO Development platform. |
| 3 | Maximum jitter induced by the secure monitor layer | 1 µs | 780 ns | Isolated executions of critical partition guarantee not exceed this value.<br>Evidences of this performance measurement can be extracted from D2.3.2 [9]. |
| 4 | Maximum overhead during the RTOS boot | 600 µs | 23 µs | Safety domains (e.g., automotive) have stringent requirements related to the RTOS boot time, which has to be completed in less than 60ms. As the secure monitor firmware adds an overhead before the RTOS execution, the goal is to setup this software layer in less than 600 µs in order to not impact the full RTOS boot time more than 1%.<br>Evidences of this performance measurement can be extracted from D2.3.2 [9]. |

| ID | KPI | Goal | Value | Comments |
|---|---|---|---|---|
| 5 | Temporal and spatial isolation by construction | Yes | Yes | Spatial isolation is guaranteed by the secure monitor firmware layer which relies on the ARM TrustZone. These evidences can be extracted from specific documentation of the secure monitor layer as well as D2.3.2 [9]. Although the current implementation gives the full priority to the RTOS, temporal isolation could also be guaranteed by the secure monitor layer, if needed. |
| 6 | Maximum latency overhead of applications inside KVM virtual machine | 5% | 1.1% | The latency overhead can be extracted from D2.2.1 [3], the experiment has been run on an ARM Chromebook with the CFS scheduler. It corresponds to the worst case scenario in term of number of workload in host and guest. |
| 7 | I/O latency inside KVM virtual machine is not affected by the I/O workload | Yes | Yes | The I/O latency is not affected by I/O workloads thanks to the V-BFQ I/O coordinated scheduler. Measurement of the I/O latency can be extracted from D2.2.1 [3]. |
| 8 | Memory bandwidth isolation by construction | Yes | Yes | Memory bandwidth isolation is guaranteed by the memguard-kvm implementation of the memguard kernel module on ARMv8 architecture. These evidences can be extracted from the D2.2.3 [4]. The implementation isolates each virtual machine regarding the executed task. |
| 9 | Memory bandwidth reservation for highest criticality level application | Yes | Yes | Memory bandwidth isolation is guaranteed by the memguard-kvm implementation of the memguard kernel module on ARMv8 architecture. These evidences can be extracted from the D2.2.3 [4]. The implementation isolates each virtual machine regarding the executed task. |

| ID | KPI | Goal | Value | Comments |
|---|---|---|---|---|
| 10 | Fault containment by construction | Yes | Yes | By leveraging on the secure monitor firmware, faults in a specific system cannot affect the correct execution of the other systems. In addition, the secure monitor firmware is able to perform a “warm reboot“ of the failed system without impacting the other systems. Specific documentation for each building block and mainly, the deliverables D2.3.2 [9] and D2.4.2 [10] could represent evidences to consider fault containment by construction. |
| 11 | Percentage of system architecture/design modelled | As much as needed for configuration file generations. | Modelling of the off-chip network and the connected nodes. | The off-chip network and the hardware of the connected nodes and the system software have been modelled to a degree that makes the execution of the tools possible. |
| 12 | Percentage of software application modelled | As much as needed for configuration file generations. | Modelling of application tasks requiring off-chip communication. | The communication related properties of the concerned tasks have been modelled, such as the size of the exchanged data and communication periods. |
| 13 | Bounded temporal network routing. (TTEthernet -> Ethernet) | <100ms | BCET: 0 ms<br>WCET: 58ms | This is defined by the period used to check the TTEthernet buffer and the time to send the message by Ethernet. The TTEthernet buffer can store up to 10 messages, which can be received with a period of 10ms. Therefore, the TTE buffer can store information until 100ms. However, it is processed each 50ms. All pending messages are processed and send through Ethernet on each check. |

| ID | KPI | Goal | Value | Comments |
|---|---|---|---|---|
| 14 | Bounded temporal network routing. (Ethernet -> TTEthernet) | <300ms | BCET: 0ms<br>WCET:100ms | The maximum transmission rate from Ethernet to TTEthernet network is limited by the period defined in the TTEthernet configuration. In the demonstrator, for the path from Ethernet to TTEthernet, the TTE configuration defines one TT slot with period 10ms and a buffer of 10 messages. These temporal requirements were based on KPI #22 to #31. The BCET is obtained when the message is received just before start the TT slot. The WCET is obtained when the message received is the latest in the buffer and it arrives just at the end of the TT slot. |
| 15 | Bounded temporal interference (network) | Safety related communications are not affected by non-safety communications | No delay introduced in safety communication when non-safety traffic is also used | Safety-related communication uses Time-Triggered network traffic type, while videos streaming use Best-Effort traffic type (i.e, normal Ethernet). Therefore, the Time-Triggered aware hardwares (TTswitch and TTethernet card) ensure that no delay is introduced when using non-safety network traffics. |
| 16 | Bounded temporal interference (processing) | | Not evaluated. | The safety related partition of the Hospital server (Juno board) run a Real-Time operating system (FreeRTOS) on which the drivers of the TTEthernet PCI card are not available. Doing the porting was resource consuming and out of the scope of DREAMS project, so, as a workaround, the PCI card is used on the Linux side sharing its data with the RT-OS through a shared memory mechanism. |
| 17 | Bounded temporal interference (resources access rate) | | Not evaluated. | See #16. |

| ID | KPI | Goal | Value | Comments |
|---|---|---|---|---|
| 18 | ST Body gateway-to-partition latency | Below 1sec for 8 (ST Bodygateway devices) x 256 samples (per ST Bodygateway) | An alternative way is to examine delay to process 256 samples on the server. We have seen that it's possible to support 1 ST Bodygateway at full rate (256 point/sec) with delay < 1sec on Zedboard (see Section 2.3, Figure 9) and up to 4 ST Bodygateways on the final healthcare demonstrator (see KPI #36 and #37). | Since the full platform is a distributed embedded system without a global synchronized clock, we examine whether the update rhythm at the hospital media server is sufficient to guarantee real-time visualization for a number of pulse sensor devices. An alternative approach, taken in Section 2 concentrates on the total delay to process 256 samples (or more) on the server (single clock reference). Both techniques provide practical approximations, since precise estimation of the proposed KPI requires modifying the final demonstrator to support clock synchronization which is beyond the project goals. |
| 19 | Percentage of development steps covered by tools in demonstrator | As many as possible. | 65% | The applicable tool chain Use Case 1 has 14 steps. The 5 steps related to on-chip communication and task scheduling configurations have not been covered since the used technologies are not supported by the tools of the toolchain. |
| 20 | Percentage of automatically executable transformations | As many as possible of the steps that can be automated. | 100% | Among the development steps covered by the tool chain, only the off-chip scheduling related steps can be automated. The exchange of data with configuration file generator TTE-Plan has been completely automated. |
| 21 | Adaptability to evolution of product and standards | To support FDA and EC standards | Yes, using a flexible scalable hardware and software platform is possible to go through the approval processes and procedures requested for FDA or EC. | Some of the standards are EN 60601, EN 62304, EN 980, EN 60529; ISO 14971, ISO 10993, EC38, EC57. |
| 22 | ST Bodygateway ECG raw data | For remote analysis | Yes, implemented | Reference: technical specification of the STM body gateway |

| ID | KPI | Goal | Value | Comments |
|---|---|---|---|---|
| 23 | ST Bodygateway Heart Rate | To calculate heart rate | Yes, implemented possible values 128 or 256 Hz | Reference: technical specification of the STM body gateway |
| 24 | ST Bodygateway Heart Rate Reliability | Heart Rate Reliability calculated using ECG raw data | Yes, implemented | Reference: technical specification of the STM body gateway |
| 25 | ST Bodygateway R-R Variability | peak Variability calculated by processing the ECG raw data | Yes, implemented | Reference: technical specification of the STM body gateway |
| 26 | ST Bodygateway BIOZ | Breathing rate calculated using the BIOZ raw Data | Yes, implemented | Reference: technical specification of the STM body gateway |
| 27 | ST Bodygateway ACC XYZ | Yes | Yes, implemented value: 50Hz | Reference: technical specification of the STM body gateway |
| 28 | ST Bodygateway Body Position | Body Posture calculated starting from the ACC info | Yes, implemented | Reference: technical specification of the STM body gateway |
| 29 | ST Bodygateway Activity level | Activity Level calculated starting from the ACC info | Yes, implemented | Reference: technical specification of the STM body gateway |
| 30 | ST Bodygateway Breathing Rate | Breathing rate calculated using the measure the bio-impedance (BIOZ) raw data | Yes, implemented | Reference: technical specification of the STM body gateway |
| 31 | ST Bodygateway Battery | Battery level measurements | Yes, implemented values to be probe 10,15,30,60,300 sec | Reference: technical specification of the STM body gateway |
| 32 | Juno R1 CPU utilization in video streaming – Maximum overhead | For 24 FPS, half-screen size, 24 bits/pixel, A57 cluster 85% on average. | 63% on average for single video, and 100% for 2 videos | Refers to average CPU utilization for video streaming on Juno R1. Video streaming virtual machine is machine is running with two virtual cores on the A57 cluster. |
| 33 | Juno R1 memory utilization in video streaming – Maximum overhead | Expected 1-2 MB/s from previous tests on Zedboard | ~5Mbit/s | Refers to average memory utilization for video streaming from Juno R1. Video streaming is allocated to two A57 cores on Juno. |

| ID | KPI | Goal | Value | Comments |
|---|---|---|---|---|
| 34 | Juno R1 – STM32F746G-DISCO Ethernet network utilization in video streaming – Maximum overhead | For 24 FPS, half-screen size, 24 bits/pixel, 80 Mbits/sec (10 MB/sec) | ~2.22 MB/sec | The network bandwidth used by the whole virtual machine streaming a video has been measured at 2.22 MB/sec. |
| 35 | STM32F746G-DISCO CPU utilization in video streaming – Maximum overhead | Expected to be less than 100% since no transcoding is used | not precisely measured, but less than 80% | CPU utilization on STM32F7 board is significantly less than 100%. However, STM32F7 runs a single application (video player), so this KPI is not so relevant (it does not affect ECG parameters). |
| 36 | Real-time characteristics of ECG Processing application | The ECG data analysis is performed in real-time. | Real-time visualization and cardiac disease detection is performed by the Hospital Server application for up to 4 ECG devices | Although the number of supported device on the full demonstrator is up to 6 (see KPI #37 below). The visualization application that also detects cardiac anomalies on the ECG is only able to handle 4 devices at the same time; with more devices the refresh rate of the visualization is too low. It is also important to note that the Hospital Server used in this demonstrator is low power server (Juno r1) compared to a machine that can be used in a real deployment.<br><br>See Appendix 6.1 for further details. |
| 37 | Scalability of the healthcare architecture in terms of number of Body Gateway devices | Up to 6 Body Gateway devices per DREAMS Harmonized Platform. | The demonstrator contains one DHP and is able to handle up to 6 Body Gateway devices. However, the system is scalable accordingly the number of DHPs | 6 Body Gateway devices are supported by the demonstrator. But only 4 of them can be visualized simultaneously on the Hospital Server, due to computing power limitation. This means that if a DHP is used per hospital room, 6 patients can be supported per room, 4 if real-time visualization on the same server is needed (considering the same hardware than in this demonstrator).<br><br>See Appendix 6.1 for further details. |

**Table 2: KPIs evaluated at M48**

## 3.2 Objectives assessment

The following tables present the progress towards the completion of measure for success and project objectives by analysing available information. The measures for success are marked with green colour if the progress is positive, orange if there is not enough information to evaluate it, and red if the progress is negative.

| Objective 1: Architectural style and modelling methods based on waistline structure of platform services | | | |
|---|---|---|---|
| Measure for success | KPIs | Evaluation | |
| 1.1 Safety | 1, 10, 22-31 | The ISO 26262 certification ASIL C of the secure monitor firmware, which corresponds to SIL-2/SIL-3 of the Functional Safety Standard IEC61508 that fits the needs of the healthcare use-case, is under functional safety assessment. The final audit is planned for mid of September, however the results of the ISO 26262 – Audit phase 1 related to the concept phase did not reveal any non-compliances. Therefore, the secure monitor firmware is able to ensure the execution of the safety features. | |
| 1.2 Real-time | 2, 3, 4, 36 | The relevant RTOS is supported and the timing requirements are met according to tests carried out in this preliminary evaluation. The ECG visualization can be also performed in real-time with up to 4 sensor devices. Therefore, real-time objectives are achieved. | |
| 1.3 Fault containment | 10 | By leveraging ARM TrustZone, the secure monitor firmware provides a system-wide security approach which isolates processor cores, bus, memory and peripherals in two separate compartments, ensuring fault containment in order to preserve the execution of mission-critical tasks if a fault occurs in the non-critical application. | |
| 1.4 Timely adaptation | 32, 33, 34, 35 | Already evaluated. Improvements relate to DHP. | |
| 1.5 Security | | | |
| 1.6 Domain-independent core services | | | |
| 1.7 System Modelling (i.e., fine grained analysis / scheduling, complexity, completeness) | 11, 12 | The DREAMS tool chain has been applied (as far as possible) to the healthcare demonstrator, including the modelling of the system (see D4.4.2). | |
| **Objective evaluation** | | | |
| All the measures for success for this objective, regarding the safety, real-time and model aspects have been evaluated positively. | | | |

**Table 3 : Objective 1 assessment**

| Objective 2: Virtualization technologies to achieve security, safety, real-time performance as well as data, safety, energy and system integrity networked multi-core chips | | |
|---|---|---|
| Measure for success | KPIs | Evaluation |

| | | | |
|---|---|---|---|
| 2.1 Isolation | 5, 6, 7, 8, 15, 16, 17 | On the gateway server (Juno), the memory bandwidth isolation is guaranteed by the memguard-kvm implementation, whereas critical applications are isolated through the secure monitor firmware relying on ARM TrustZone. On the network, the critical traffic uses Time-Triggered traffic class while non safety-related traffic (entertainment) use Best-Effort traffic class, this ensure a total isolation for the safety-related communication. | |
| 2.2 Reduced bank conflicts | | | |
| 2.3 Gateways | 13, 14 | The temporal requirements for the management of the network traffic are achieved. However, the restrictions listed on section 2.2.4 shall be taken into account when the periodic burst transmission is considered in the Odroid board. | |
| 2.4 Reduction of latencies | 6 | The co-scheduling implementation for KVM virtual machines allows minimizing the overhead. | |
| 2.5 Reduction of jitter | 18 | The latency of the whole demonstrator has been evaluated and is enough to meet the needed real-time requirements. | |
| 2.6 Reconfiguration | | | |
| 2.7 Security | 1, 5 | The secure monitor layer ensures the security configuration of ARM TrustZone in order to instantiate a secure compartment isolated from non-critical accesses. | |
| **Objective evaluation** | | | |
| All the measures for success for this objective have been evaluated positively. The isolation provided by the different technology blocks allow strong isolation, security and latency reduction. | | | |

**Table 4: Objective 2 assessment**

| Objective 3: Adaptation strategies for mixed-criticality systems to deal with unpredictable environment situations, resource fluctuations and the occurrence of faults | | | |
|---|---|---|---|
| Measure for success | KPIs | Evaluation | |
| 3.1 Variability | 5, 9 | Critical applications (e.g., bandwidth, peripheral, memory, etc) are isolated from faults which occur in other partitions. It has been tested by forcing a crash in the Normal partition running on top the secure monitor firmware, which does not impact the execution of the Secure partition containing the mission-critical tasks. | |
| 3.2 Criticality spectrum | | The architecture and technologies ensure the correct isolation of the criticality applications for the healthcare demonstrator. | |
| 3.3 Applicability | | | |
| 3.4 Efficiency | | | |
| 3.5 Scalability | 37 | The number of “Body Gateway” supported by the healthcare demonstrator is up to 6 devices, 4 if the visualization in real-time is needed. | |
| 3.6 Portability | | All technologies used in the healthcare demonstrator have been developed in other Work Packages (e.g., | |

| | | | |
|---|---|---|---|
| | | WP2). In this context, portability can be positively assessed. | |
| **Objective evaluation** | | | |
| All the measures for success for this objective have been evaluated positively. | | | |

**Table 5: Objective 3 assessment**

| Objective 4: Development methodology and tools based on model-driven engineering | | | |
|---|---|---|---|
| Measure for success | KPIs | Evaluation | |
| 4.1 Development process | | | |
| 4.2 Development steps covered by tools | 19 | The DREAMS tool chain has been applied (as far as possible) to the healthcare demonstrator, including the identification of the toolchain use case and associated development steps (see D4.4.2). | |
| 4.3 Automatically executable transformations | 20 | See #19, above. | |
| **Objective evaluation** | | | |
| All the measures for success for this objective have been evaluated positively. | | | |

**Table 6: Objective 4 assessment**

| Objective 5: Certification and mixed-criticality product lines | | | |
|---|---|---|---|
| Measure for success | KPIs | Evaluation | |
| 5.1 Modular safety-case | | | |
| 5.2 Safety-case modularity | | The Hospital server is based on the secure monitor firmware to consolidate non-critical partition along with mission-critical tasks. This software component is certified, therefore, only the critical application needs to be certified when changed. | |
| 5.3 Architectural support | | Some components used in the demonstrator are generic enough to be used in other use cases, while some components of platform specific. | |
| 5.4 Configuration optimization | | | |
| 5.5 Variability | | Critical applications (e.g., bandwidth, peripheral, memory, etc) are isolated from faults which occur in other partitions. | |
| 5.6 Domains and market features | 21 | It is possible to go through the approval process and procedures requested by FDA or EC. | |
| **Objective evaluation** | | | |
| All the measures for success for this objective have been evaluated positively. | | | |

**Table 7: Objective 5 assessment**

| Objective 6: Feasibility of DREAMS architecture in real-world scenarios | | | |
|---|---|---|---|
| Measure for success | KPIs | Evaluation | |
| 6.1 Separation | 1, 5 | According to KPI values obtained in the preliminary evaluation, the level of time and space separation obtained in the demonstrator is enough to perform certification. | |
| 6.2 Standard compliance | 1 | The secure monitor firmware is under functional safety assessment according to the ISO 26262. The | |

| | | | |
|---|---|---|---|
| | | final audit is planned for mid of September, however it is important to notice that the ISO 26262 – Audit phase 1 has been achieved without any major non-compliances raised by the auditor, thus meaning that the secure monitor firmware is on the right way to comply with the standard. | |
| 6.3 Cost | | The mixed-criticality concept applied to this demonstrator allow to combine critical applications along with non-critical ones, inside the same SoC or network. Therefore, less hardware is required, lowering the cost. Although, in this demonstrator only development platforms are used which have a very high cost compared to production hardware. | |
| 6.4 Reusability | | Most of the components, not directly related to healthcare, can be used in other domains where mixed-criticality is needed. | |
| 6.5 Extensibility | | Most of the technology block and component can be extended to cover other use cases. | |
| **Objective evaluation** | | | |
| All the measures for success for this objective have been evaluated positively. The implementation done for this demonstrator can be performed in real-world scenarios related to healthcare domain. | | | |

**Table 8: Objective 6 assessment**

| Objective 7: Promoting widespread adoption and community building | | | |
|---|---|---|---|
| Measure for success | KPIs | Evaluation | |
| 7.1 Community infrastructure | | All technologies used in the healthcare demonstrator have been developed in other DREAMS Work Package. Most of these technologies have been exposed though the DREAMS project, and some components have been open-sourced (e.g., KVM modifications, MemGuard, …) via the Mixed Criticality Forum website. | |
| 7.2 Training material | | All technologies used in the healthcare demonstrator have been developed in other DREAMS Work Package. Most of these technologies (e.g., KVM) have been presented in video training session available on DREAMS YouTube channel. In this context, the measure for success can be positively assessed. | |
| 7.3 Standardization | | The secure monitor firmware is certifiable up to ASIL C according to ISO26262 which corresponds to SIL2/3 of IEC61508. | |
| 7.4 Roadmap | | | |
| **Objective evaluation** | | | |
| All the measures for success for this objective have been evaluated positively. | | | |

**Table 9: Objective 7 assessment**

# 4 Conclusion

In this document, the result of the assessment of the technological results of the project within the Healthcare demonstrator is reported, including the use case and scenarios as well as the technological results. Key Performance Indicators have been calculated or estimated in order to provide measurable, quantitative and objective information to evaluate the measures for success and the achievement of the objectives.

Section 2 presents the current status of the demonstrator, on which all the hardware and software components are integrated, including the DREAMS harmonized platform. The demonstrator aims to replicate a hospital use case where critical and non-critical data are routed among a network of heterogeneous platform, allowing sensitive ECG data to be used by a medical staff while media content is streamed to patients at the same time. The isolation of the critical data inside this mixed-critical demonstrator is achieved thanks to three main technologies, with XtratuM on the DREAMS Harmonized Platform, the Time-Triggered network class and KVM with the secure monitor firmware on the Juno board.

The assessment results were provided in Section 0. All the KPIs are considered, but the ones that can only be objectively evaluated after the project (marked with an **A**) were only estimated. Overall, KPIs are evaluated with success, as well as the measures for success and the objectives of the project. The main goals of the healthcare demonstrator have been satisfied. Non-safety-critical data and applications have been integrated along with safety-critical ones without affecting them, creating a mixed-criticality system.

# 5 Bibliography


[1] D8.2.1 - System Demonstrator running mixed-criticality healthcare and entertainment use case, DREAMS Consortium, 6/2016

[2] D8.3.1 - Preliminary assessment report related to improving or calibrating the technological results, DREAMS Consortium, 2017

[3] D2.2.1 - Optimized hierarchical real-time scheduling heuristics at the network interface layer and their seamless integration into a real-time KVM hypervisor, DREAMS Consortium, 3/2015

[4] D2.2.3 - Implementation of real-time scheduling heuristics and coordination for the KVM hypervisor, DREAMS Consortium, 4/2016

[5] DREAMS, Distributed Real-Time Architecture for Mixed-Criticality Systems: Description of Work, in DOW2014. p. 260.

[6] ISO 26262 - Part 4: Road vehicles - Functional Safety - Product development at the system level

[7] ISO 26262 - Part 6: Road vehicles - Functional Safety - Product development at the software level

[8] ISO 26262 - Part 8: Road vehicles - Functional Safety - Supporting processes

[9] D2.3.2 – Firmware monitor layer implementation for the concurrent execution of an RTOS and Linux/KVM, DREAMS Consortium, 7/2016

[10] D2.4.2 - Extensions and modifications related to supporting integration with industrial demonstrators, DREAMS Consortium, 7/2016

[11] M. Hadjem, O. Salem, and F. Nait-Abdesselam, "An ECG monitoring system for prediction of cardiac anomalies using WBAN", *in Proc. Conf. e-Health Netw. Appl. and Services*, 2014, pp. 441—436.

[12] J. Ko, J. H. Lim, Y. Chen, R. Musvaloiu-E, A. Terzis et al., "Medisn: Medical emergency detection in sensor networks", *ACM Trans. on Embedded Comput. Syst.*, 10 (1), p. 11, 2010.

[13] J.A. Walsh III, E.J. Topol, S.R. Steinhubl. 2014. “Novel wireless devices for cardiac monitoring”, *New Drugs and Technologies*, pp. 573—581.

[14] [s4,2] American Heart Organization, https://www.heart.org/idc/groups/ahamah-public/@wcm/@sop/@smd/documents/downloadable/ucm_491265.pdf

[15] Alivecor, https://www.alivecor.com/

[16] L.A. Saxon, "Ubiquitous wireless ECG recording: a powerful tool physicians should embrace", *J. Cardiovascular Electrophysiology*, **24 (4)**, pp. 480–483, 2013.

[17] BG Heart, http://www.preventicesolutions.com/services/body-guardian-heart.html

[18] Lifemonitor, http://www.equivital.co.uk/products/tnr/sense-and-transmit

[19] NowCardio, https://contex-tech.com/medical/nowcardio

[20] Physiomem, http://www.getemed.net/en/telemonitoring/physiomemr-pm-1000

[21] S. Gradl, P. Kugler, C. Lohmüller, and B. Eskofier, "Real-time ECG monitoring and arrhythmia detection using Android-based mobile devices", *in Proc. Conf. IEEE Engin. Medicine and Biology Society*, 2012, pp. 2452--2455.

[22] J.J. Oresko, Z. Jin, J. Cheng, S. Huang, et al., "A wearable smartphone-based platform for real-time cardiovascular disease detection via electrocardiogram processing", *IEEE Trans. Info Tech. Biomedicine*, **14 (3)**, pp. 734–740, 2010.

[23] T.-H. Yen, C.-Y. Chang, S.-N. Yu, "A portable real-time ECG recognition system based on smartphone", *in Proc. Conf. IEEE Engin. Medicine and Biology Society*, 2013, pp. 2=7262—7265.

[24] S. Hu, H Wei, and Y. Chen, " A real-time cardiac arrhythmia classification system with wearable sensor network", *Sensors*, **12**, 2012, pp. 12844—12869.

[25] A.M. Patel, P.K. Gakare, and A.N. Cheeran, "Real-time ECG feature extraction and arrhythmia detection on mobile platform", *J. Comp. Appl.*, **44 (23)**, 2012, pp 40—45.

[26] J. Weng, X.M. Guo, L.S. Chen, Z.H. Yuan et al., "Study on real-time monitoring technique for cardiac arrhythmia based on smartphone", *J. Medical and Biological Engineering*, **33 (4)**, pp.394—399.

[27] A. Iglesias, R. Istepanian, J.G. Moros. "Enhanced real-time ECG coder for packetized telecardiology applications", *IEEE Trans. Info Tech. Biomedicine*, *10 (2)*, 2006, pp. 229—236.

[28] J.D.-Ferrer, D. Sánchez, G.R.-Torrell, "Anonymization of nominal data based on semantic marginality", *Info Sciences*, **242**, 2013, pp. 35--48.

[29] P.S. Hamilton, S. Patrick, and W.J. Tompkins. 1986. "Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database", *IEEE Trans. Biomedical Engin.*, **12**, pp. 1157—1165.

[30] W.J. Tompkins. 1985. "A real-time QRS detection algorithm", *IEEE Trans. on Biomedical Engin.*, **3**, pp. 230--236.

[31] Soft Real Time ECG Analysis and Visualization, https://physionet.org/works

[32] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano, "Secure memory accesses on networks-on-chip," *IEEE Trans. Comput.*, **vol. 57, no. 9**, pp. 1216–1229, Sep. 2008.

[33] J. Porquet, A. Greiner, and C. Schwarz, "NoC-MPU: A secure architecture for flexible co-hosting on shared memory MPSoCs," *in Proc. Design Autom. Test Europe*, Grenoble, France, Jul. 2011, pp. 591–594.

[34] A. Wiggins, S. Winwood, H. Tuch, and G. Heiser, "Legba: Fast hardware support for fine-grained protection," *in Proc. 8th Asia-Pac. Conf. Adv. Comput. Syst. Archit.*, Aizuwakamatsu, Japan, 2003, pp. 320–336.

[35] M.D. Grammatikakis, K. Papadimitriou, P. Petrakis, A. Papagrigoriou, G. Kornaros, I. Christoforakis, O. Tomoutzoglou, G. Tsamis, and M. Coppola, "Security in MPSoCs: A NoC Firewall and an Evaluation Framework", *IEEE Trans. Computer-Aided Design*, **34(8)**, pp. 1344-1357, Aug, 2015.

# 6 Appendix

## 6.1 Additional information related to KPIs 36 and 37

Our default configuration for Juno (ARM v8 architecture) assigns 2 ARM A57s to the two Video virtual machines (ARM v8), and 4 ARM A53s to the ECG virtual machine (ARM v7, identical architecture to the one described in Section 2). Since in our experiments we are limited to two video streams (CPU bandwidth 100%), for each scenario, we consider 3 cases: (i) ECG in isolation, (ii) ECG with one video stream, and (iii) ECG with two video streams. Moreover, since a limited number of video streams does not significantly affect memory bandwidth (1-2 MB/s increase), we have not configured MemGuard/NetGuard LKMs on ARM Juno.

Notice that using ARMv8 64bits VM for running ECG on Juno is not feasible at all due to limited support of 32-bit compatibility libraries on ARM v8 which are required for running xview toolkit. The xview toolkit is an X11 utility (360k lines of code) on top of which our fast WAVE ECG visualization application is built, however xview has not been ported yet on 64-bit architecture.

Figure 17 shows the display on the Hospital Server when a single BodyGateway transmits an ECG via the full platform (DHP with XtratuM, TTE, Juno). This test performed successfully on the full platform with 0, 1, and 2 videos. Very small performance degradation (in the display) appears when we simultaneously use video streaming (see also Table 10).

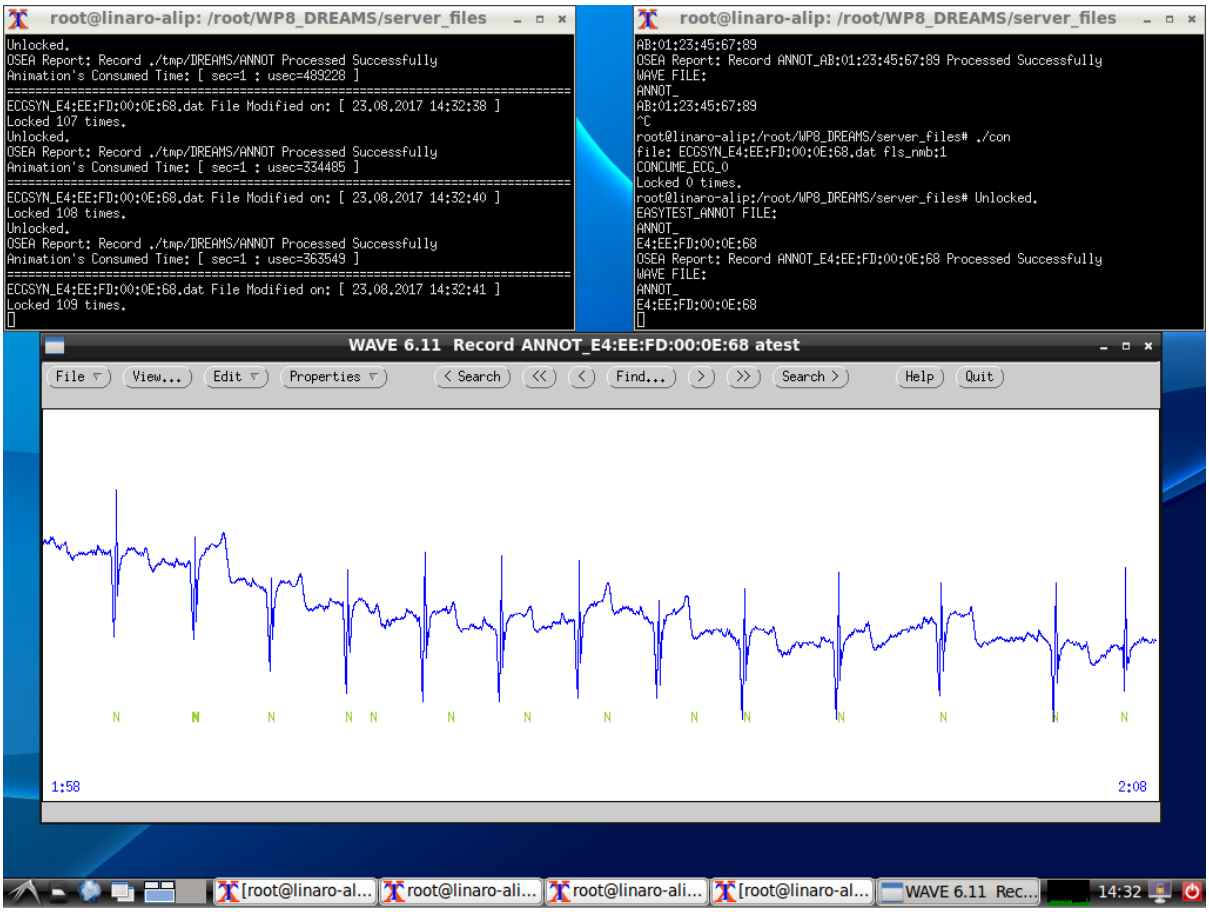


**Figure 17: ECG signal for one ST Bodygateway on full platform**

To test the scalability of the platform a “fake ECG generator” that generates fake ECG data was also used. It allows to test the platform with as many “devices” as needed.

In order to check correctness and estimate bounds to scalability we have run the test with multiple fake ECGs (1 to 16) not in the full platform, but only on Juno (without DHP, TTE). The test worked correctly and the approximate rhythm of updates is shown in Table 10. Notice that both the average and range of values increase linearly with the number of ECG signals (or patients). Since we are capable to visualize up to 10 seconds of the ECG signal in a single WAVE application screen, the update rate is sufficient to support soft real-time for up to 4 ECGs.

| Number of ECGs | Typical Update Rhythm for WAVE visualization (sec) |
|---|---|
| 2 | 3-5 |
| 4 | 7-12 |
| 8 | 15-25 |
| 16 | 31-54 |

**Table 10: Update rate for WAVE visualization without video streaming; the update rate is not much affected for 1 or 2 video streams and less than 8 ECGs, since video streams run on separate processors and memory bandwidth requirements are minimal 1-2MB/s; for 16 ECGs and two videos the update rate increases slightly to 39 to 65 secs.**

The same test has been performed considering the full platform, but only up to 6 “fake ECG” devices. Results are similar to the ones run in the Juno only (Table 10).

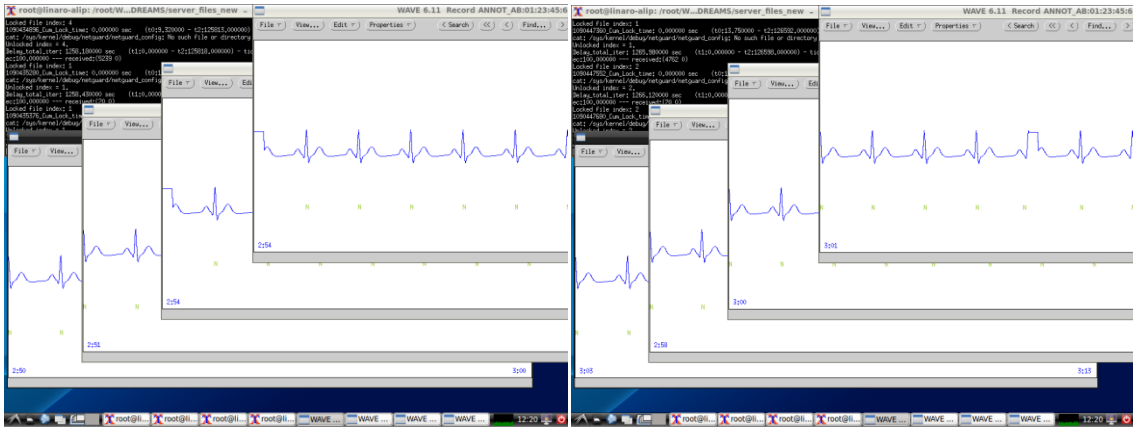

**Figure 18: Delays during successive updates on Juno (average 6-7 sec, worst-case 13).**

Figure 18 shows two successive updates (snapshots) of the 4 wave processes via wave-remote corresponding to 4 fake ECG traffic patterns. ECG analysis runs on the Juno. This case corresponds to emulating 4 ST Bodygateways and can be considered as a limit to scalability in terms of available processing power, since the application is able to capture the heartbeats of successive computations almost all the time. Furthermore, notice that for 8 emulated ST Bodygateways, delays become much larger (15 to 25s), and ECG signals cannot be updated by the display application in real-time (i.e. most points will never appear on the screen).