Distributed Real-time Architecture for Mixed Criticality Systems

Preliminary assessment report related to improving or calibrating the technological results
D 8.3.1

Project Acronym	DREAMS	**Grant Agreement Number**		FP7-ICT-2013.3.4-610640	
Document Version	2.0	**Date**	2017-01-03	**Deliverable No.**	D 8.3.1
Contact Person	Kevin Chappuis	**Organisation**		Virtual Open Systems	
Phone	+33663254852	**E-Mail**		k.chappuis@ virtualopensystems.com	

Contributors

Name	Partner
Alexander Spyridakis	VOSYS
Kevin Chappuis	VOSYS
Jorn Migge	RTAW
Miltos Grammatikakis	TEI
Arjan Geven	TTT
Marcelo Coppola	ST
Donatus Weber	USIEGEN
Hamidreza Ahmadian	USIEGEN

Table of Contents

1 Introduction

This document is the deliverable D8.3.1 of the DREAMS project. It is the first deliverable of task *T8.3 – Project technologies assessment* of work package *WP8 – Healthcare Use Case and Demonstrator*. This deliverable, *D8.3.1 – Preliminary assessment report relate to improving or calibrating the technological results,* presents the report of the first activities that have taken place with respect to the initial assessment of the DREAMS technological results targeting the Healthcare demonstrator.

In this document the overall status of the healthcare demonstrator is presented, the importance and interactions of current technological results, as well as how these are deviating from the initial project view and in what way they can be further improved.

1.1 Position of the Deliverable in the Project

The goal of work package WP8 is to develop a system platform demonstrator, integrating technologies developed in WP1 – 5 in order to assess the mixed-criticality approach in DREAMS with a complex use case from the healthcare domain. The WP8 work plan consists of three stages: Development of the architecture and specification of the use cases, System platform implementation and platform integration with technological results, and Validation/ evaluation of the platform and the project approach.

As a result, WP8 aims at the following objectives:

- To validate the DREAMS approach by implementing a realistic demonstrator executing use cases with mixed safety and performance requirements.
- To provide qualitative and quantitative measurements of technologies developed in research work packages towards design and implementation of mixed-criticality embedded systems.
- To estimate cost-effectiveness of platform design and reduction of time-to-market.
- To pave the way to exploitation by using real applications developed as show cases to expose advantages of the DREAMS approach.

The implementation of the demonstrator and the assessment is supported by the partners from the technology WPs using the respective support tasks (T1.8, T2.4, T3.4 and T4.4).

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.

1.2 Contents of the Deliverable

In chapter 2, we provide the current status of the Healthcare demonstrator platform, while in chapter 3 the overall approach to the use case and scenarios is presented. Chapter 4 consists of the technological results included in the demonstrator, together with their assessment and future projections towards the final state of the demonstrator. Conclusions and next steps are finally presented in chapter 6.

2 Hardware platform

In this chapter the overall hardware architecture involved in the WP8 healthcare demonstrator is detailed, along with the current status of each hardware component and its role in the demonstrator.

2.1 Overview

The current form of the Healthcare demonstrator relies on two main hardware sets: client, server sets. From one side we have the client set that includes the DREAMS harmonized platform (DHP), the ST body gateways and STM32 smart video displays, while on the server side the Juno platform takes the role of the Hospital Gateway. These two sets are connected with Time-Triggered Ethernet (TTE) through a TTE switch, providing accurate timing and synchronization features for the communication between the two end points. In particular the connection between the TTE switch and STM32 smart video displays are using the Best effort connectivity.

As it will be also further detailed in the next chapter, regarding the Healthcare use case, the hardware components in the demonstrator are complemented with additional platforms, which aim to ease integration and bolster needed features that might not be applicable on the two main devices. One example of this, is the usage of an ODROID-XU ARMv7 platform with the purpose of executing the Bluetooth protocol (BT) stack , which is not straightforward to do so in the case of XtratuM on the DHP. Instead of redeveloping the BT stack directly from the XtratuM partitions the action is offloaded to a remote target, since this is beyond the scope of the DREAMS project.

The targeted architecture for the hardware components of the demonstrator is ARM, specifically ARMv7-A and ARMv8-A. DHP is using the 32-bit ARMv7 architecture with Cortex-A9 processors, whereas the Juno board is using the 64/32-bit ARMv8 architecture with Cortex-A57/A3 processors. This choice covers the latest ARM processors, but also the popular Cortex-A9, which is considered a good compromise between features, power efficiency and performance.

2.2 Demonstrator components

For the current version of the demonstrator, a more detailed look is provided for the DREAMS Harmonized platform as well as the Juno board. Other hardware components that are planned to be used in the demonstrator will not be covered in this chapter.

2.2.1 Dreams harmonized platform

The DREAMS harmonized platform consists of a Xilinx Zynq-7000 AP SoC ZC706 Evaluation board (Figure 1). It constitutes a homogenous basis for the integration of respective DREAMS technological building blocks and bundles the integration efforts of the technology providing partners for the three DREAMS demonstrators in work packages WP6 (Avionics), WP7 (Wind Power) and WP8 (Healthcare).

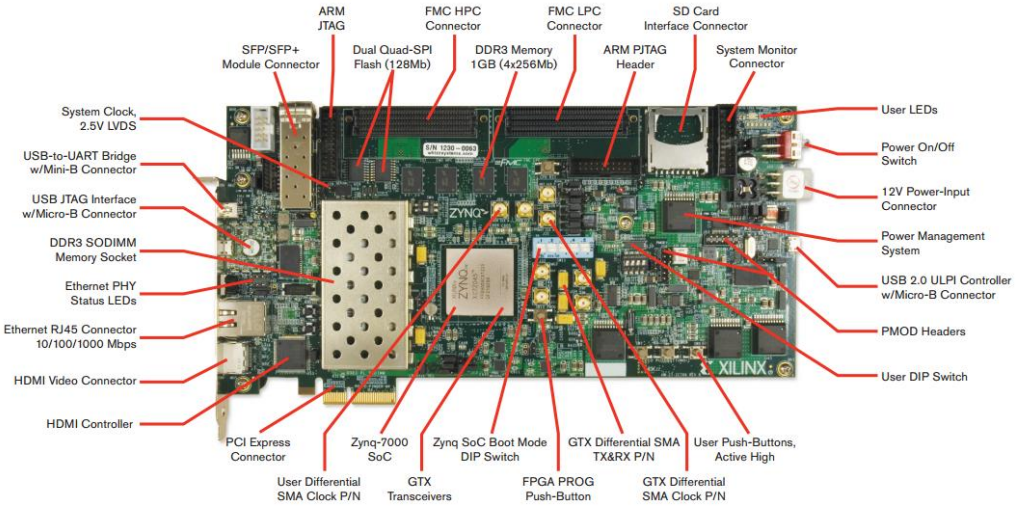

Figure 1: Xilinx ZYNQ ZC706 evaluation board with ZYNQ 7000 SoC [2]

The ZYNQ 7000 SoC combines the software programmability of a processor (ARM A9 dual core processor, ARMv7 architecture, Processing System (PS)) with the hardware programmability of an FPGA (XC7Z045, Kintex-7, Programmable Logic (PL)). Two DDR memories (1GB each) are available on the board one of them attached to the PL, the other one attached to the PS.

The following technological building blocks provide the core functionality of the DHP:

- STNoC for chip level communication
- XTRATUM as virtualization solution for the ARM A9 dual core
- TTE gateway for cluster level communication

Figure 2 gives an overview on the logical blocks and communication paths between the components of the DHP. Main connecting element is the adapted STNoC and the associated network interfaces that provide support for time-triggered (TT), rate-constraint (RC) and best-effort (BE) traffic. Two virtual networks VN1 and VN2 establish communication channels between the components attached to the STNoC. The two virtual networks serve two different priority levels:

- VN1, high priority, red color
- VN2, low priority, blue color

Beside the dual core ARM A9, there are three additional µBlaze processing cores available (µB0, µB1, µB2). The cluster level interface is provided by the attached TTE controller acting as on/off chip gateway. Access to the DDR memory of the PL is provided by the DDR controller connected to the STNoC. The unit serves as memory tile for the DHP node. Furthermore, an Accelerator Coherency Port (ACP) allows the access of optional hardware units.

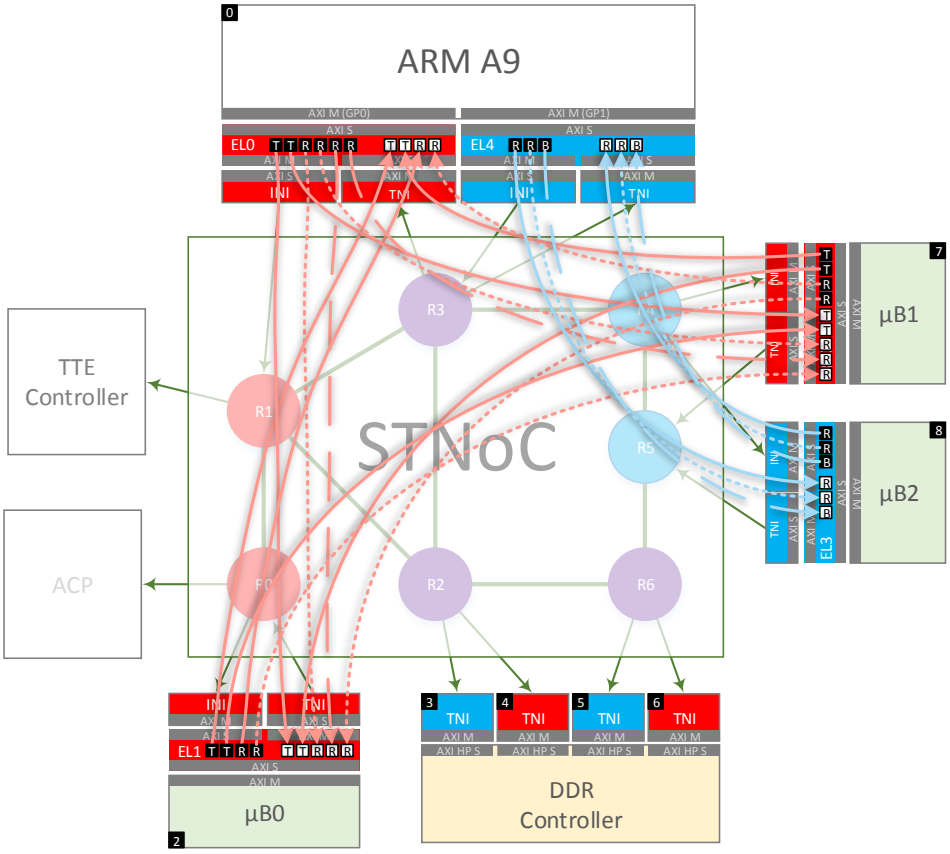

Figure 2: DREAMS harmonized platform logical blocks and communication paths

More detailed information on the DREAMS harmonized platform is available in the dedicated deliverable document D1.5.1 [1].

2.2.2 ST Body Gateway

Figure 3: ST Body Gateway device

The ST BodyGateway Device is a wearable electronic, battery operated device that is worn on the chest for the acquisition, recording and transmission of physiological parameters to external devices which can analyze or forward the data to additional storage elements or system.

The ST Bodygateway device is also capable to record symptomatic and asymptomatic events and is indicated for ambulatory monitoring of non lethal cardiac arrhythmias.

Additionally, resident in this device is a heart rate, respiration rate and activity level calculation algorithm, which allows the system to manage information messages from/to the Server according to specific settings defined by the physicians/operators. In summary the list of possible medical parameters are:

- Heart rate
- Heart rate reliability
- RR Interval variability
- Breathing rate
- Activity level
- Body position

The device is a part of a Multi-parameter Analysis System, the ST Body Gateway communicates via a BT radio link with the external device. Specification of ST Body Gateway is beyond the scope of this document. At its heart of this device we have a ST Microelectronics STM32 (32-bit ARM Cortex microcontroller with embedded Flash), chosen for its flexible architecture and low power processing capability. Bluetooth radio was selected for connectivity because its availability in most commercial solutions to ensure proper coverage and patient access.

2.2.3 The STM32 smart display

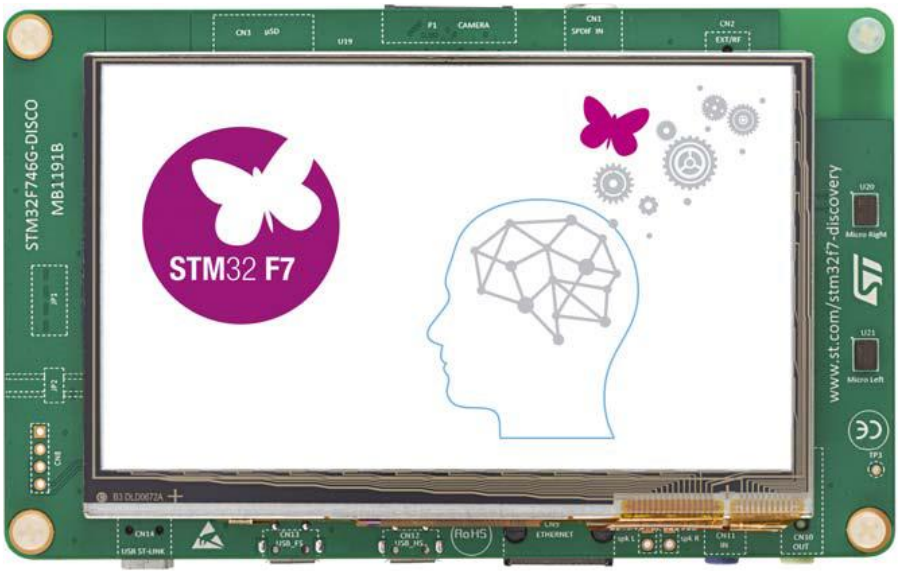

Figure 4: STM32 smart display

The STM32F746G-DISCO discovery board is a complete demonstration and development platform for STMicroelectronics ARM® Cortex®-M7 corebased STM32F746NGH6 microcontroller. This microcontroller features four I2Cs, six SPIs with three multiplexed simplex I2S, SDMMC, four USARTs, four UARTs, two CANs, three 12-bit ADCs, two 12-bit DACs, two SAIs, 8- to 14-bit digital camera module interface, internal 320+16+4-Kbyte SRAM and 1-Mbyte Flash memory, USB HS OTG, USB FS OTG, Ethernet MAC, FMC interface, Quad-SPI interface, SWD debugging support.

The full range of hardware features on the board helps users to evaluate almost all peripherals (USB OTG HS, USB OTG FS, 10/100-Mbit Ethernet, microSD™ card, USART, SAI Audio DAC stereo with audio jack input and output, MEMS digital microphones, SDRAM, Quad-SPI Flash memory, 4.3-inch color LCD-TFT with a capacitive multi-touch panel, SPDIF RCA input, etc.) make it possible to easily use in the Healthcare demonstrator to connect the TTE switch.

2.2.4 Juno ARM development platform

The Juno development board is the first ARMv8-A reference platform with its initial version being released by ARM in the second half of 2014. The Juno System-on-Chip comes with a range of IPs that are directly used by DREAMS, related to network and cache-coherent interconnect, TrustZone and the Cortex-A57/A53 processors.

In more detail this is the list of the major features included with Juno:

- AArch64 and AArch32 architecture (64-bit support and 32-bit backwards compatible)
- Cortex-A57 (2 cores) and Cortex-A53 (4 cores) clusters in big.LITTLE topology – 8GB of RAM
- PCI-express 2.0 (revision 1 and 2)
- Various peripherals: HDMI, USB 2.0, Gigabit Ethernet, DMA, SATA, etc.
- TrustZone Memory Controller fully supporting the security extensions of the processors
- Mali T624 GPU
- Virtualization Extensions

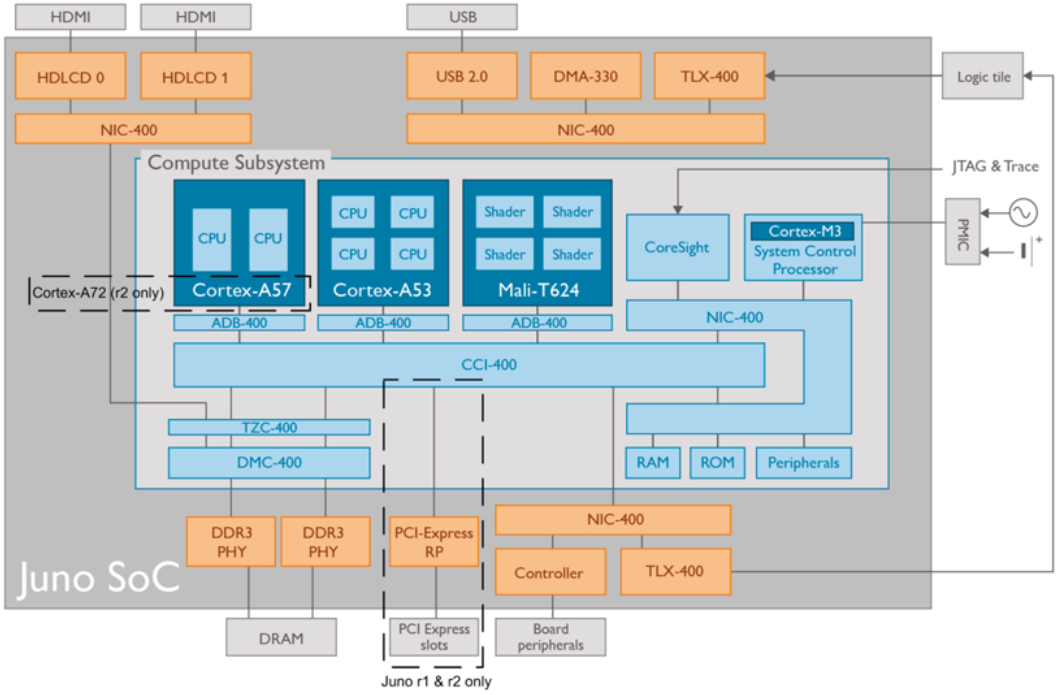

Figure 5: Juno architecture overview

In the context of DREAMS, the Juno platform offers significant flexibility in terms of development and prototyping features. The most important characteristic is that the latest ARMv8-A architecture enables 64-bit computing for embedded systems, while preserving backwards compatibility with ARMv7 payloads. This allows to run legacy software both for normal host applications but also when using the virtualization features of the platform. Coupled with Linux/KVM, this translates to the ability of executing seamlessly both 64 and 32-bit guests.

In terms of security and isolation, Juno is fully integrated with TrustZone and the security extensions found on most ARM processors. This is achieved by including the TZC-400 IP, which allows the separation of resources in secure and normal worlds. In DREAMS and the healthcare demonstrator, this feature is used to implement a secure monitor firmware layer, which enables the concurrent execution of an RTOS with Linux/KVM and ensuring hardware isolation between the two.

By combining the virtualization and security extensions of the platform, almost any workload can be executed on Juno, ranging from hard real-time tasks and legacy software, to feature-rich multimedia

environments and even light server and networking workloads. This sort of flexibility makes the Juno board an ideal candidate for prototyping a proof of concept of the healthcare demonstrator.

For the communication needs of the demonstrator the Juno will be coupled with a TTEthernet device by utilizing the integrated PCIe bus of the board. This will ease the integration process to achieve the final results of directly connected the DHP and Juno. In this context, it is worth mentioning the capability of the Juno board to be coupled with an FPGA daughterboard (LogicTile). The FPGA is connected directly to the ARM SoC through the AXI bus, providing even more prototyping capabilities. This approach could be an alternative for attaching a TTEthernet device, though for integration purposes the PCIe bus seems a more feasible target in the context of DREAMS.

In essence, the Juno development platform is a first reference implementation for ARMv8 devices, and its purpose is quite fitting with the Healthcare demonstrator use case as a Hospital Gateway. As seen by the latest ARMv8 platforms, for the automotive and server markets (e.g. Renesas' R-CarH3 and Cavium's ThunderX), a lot of the features found on Juno are now included in these latest products.

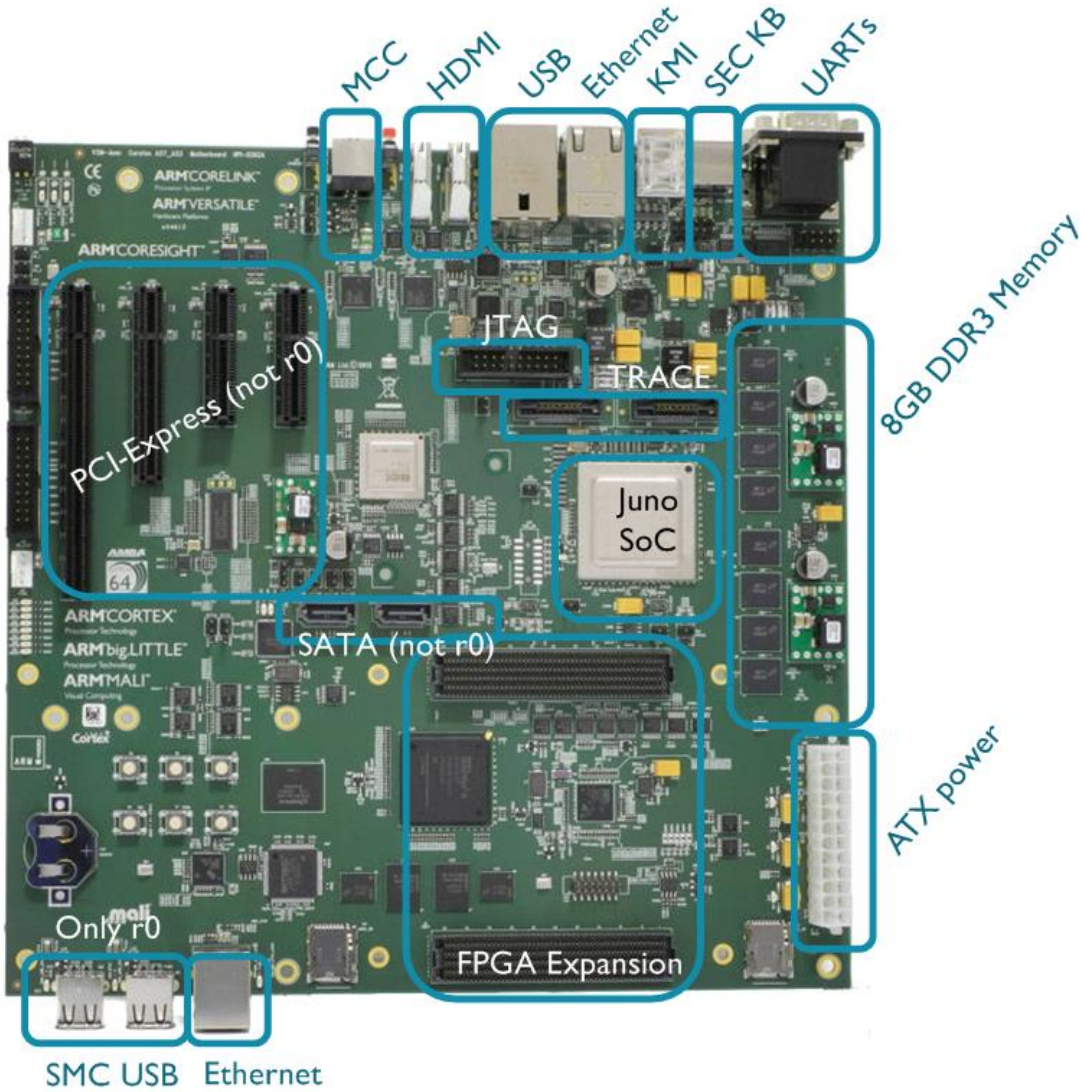

Figure 6: Juno motherboard overview [3]

3 Healthcare use case

For the purpose of a unified demonstrator with the DREAMS platforms and various technological results, the Healthcare use case scenario is covered in this chapter. The use case includes interactions between the DHP as a client with a number of XtratuM partitions, together with the Juno development platform acting as the Hospital Gateway on which Linux/KVM is running concurrently with a hardware isolated RTOS. In the following sections the top level use case is described together with the scenarios used to implement the demonstrator.

3.1 Top level use case description

An integrated flexible system for screening, prevention and management of disease will be the target in the Healthcare use case. This use case involves the streaming of premium and non-premium video content to several patients located in different rooms. Currently they are restricted to see standard Over-the-top (OTT) content (no premium) via a Set-top-box (STB), in addition the monitoring of the overall health and well-being by using Body Gateway devices is realized in stand-alone mode. In the Healthcare DREAMS Use Case, we introduce the Hospital Media Gateway

- To enable premium and non-premium content consumption by a whole of TV sets (using a wired network) without the need for a dedicated per-device STB.
- To enable the remote monitoring by interconnecting all the Body Gateways systems
- To have a single shared network supporting mixed critical traffic

This use case enables continuous or intermittent physiological monitoring and detection of abnormalities arising from a range of medical conditions coming from different patients. As matter of fact, when the patient has a clinically relevant event the Body Gateway communicate wirelessly with a central wireless switch that can be located in each hospital room and finally with the Hospital Media Gateway. The information sent by the Body gateways are examined by the patient's caregiver or doctor for the necessary intervention. At the same time this technology can provide a valuable real-time feedback to the person wearing the body monitoring. Patients' data are logged via a dashboard and warnings are sent in real-time. This healthcare dashboard will be executed in the execution environment of the Hospital Media Gateway where other less critical applications, such as the premium content consumption are concurrently executed.

Although the DREAMS use case targets multiple rooms, to reduce the complexity and the underline cost of devices we target to a single room scenario. This use case involves the collection of medical data from 2 patients collocated in the same room in a Hospital. The collection of medical data is performed using 2 Body Gateway units. Once the data has been collected, the Body Gateway Control Units can stream directly to a Hospital Media Gateway through the wireless network as illustrated in Figure 5.

Figure 7: Healthcare demonstrator main use case

On the other side, the Hospital Media Gateway is receiving the Body Gateway data using the wired connection that in the case of DREAMS is TTTEthernet.

In addition the Hospital Media Gateway is also the termination of a variety of DRM and CAS protection schemes, therefore needs to convert them to a single common link protection mechanism that can be easy transmitted via TTEthernet towards the simple hubs located in each room. The hub on top of the bridging functionality between the Bluetooth and TTTEthernet, it may implement one or more of the following items

- Content transcoding to adapt the content format to suit each receiving device, e.g. to match the screen resolution, or frame rate.
- To implement the security checks (e.g. each device type is allowed to display the content under the terms of the license agreements)
- To provide a local content-rendering capability to allow direct wired connection via HDMI
- To unprotect the content to be visualized

In the aforementioned use case we need to address three technical requirements. The first one, involves ensuring real-time data transmissions. This implies that we need to guarantee real-time operations of the overall distributed system, which comprises of different subsystems with different criticality levels that are sharing the network and computational resources. This implies that the healthcare architecture should provide guarantees in presence of criticalities among the computation components (Server, Body Gateway,) and the communication infrastructure (wired and wireless network). Another important requirement is the potential scalability of the healthcare architecture since the number of Body Gateway devices can be scaled up and down depending on the number of patients to be monitored. This implies that system components can vary in time as components are added or removed while the system is running. These changes should not affect any property (e.g. real-time requirement) of the system. Last but not least is reliability requirement in presence of faults. The healthcare systems and the related use cases are networked systems that share information, monitor performance and enhance safety. In addition healthcare systems demand more cost-efficient electronic components with smaller footprints for space-constrained applications, thereby requiring more integration and performance enhancements from semiconductors.

3.2 Healthcare Scenarios: ECG Diagnosis and Security

Figure 8: The Body Gateway pulse sensor device

A typical hospital gateway logs biometric patient data transmitted from different devices, such as pulse sensors. For example, the STMicroelectronics' *Body Gateway* (shown in Figure 6) is a mobile, flexible, body-worn cardiac monitoring sensor that allows physicians to monitor important biometric patient data (e.g. heart rate (1-lead ECG), respiration rate, activity level, body position) while patients remain active and independent. The device acquires, digitalizes and either streams in real-time via a Bluetooth radio link (or stores and periodically transmits) physiological data to a professional physician's mobile phone or server in order to access patient data securely anytime and review alerts, such as sudden arrhythmia or cardiac arrest. This cutting edge technology can be used to track and support constantly, elderly people, home monitoring, chronic cardiac disease monitoring, or 24-hr holter ECG.

The main objectives of this work carried out by TEI (within WP2 Deliverables D2.4.1 and D 2.4.2) with support from ST are as follows.

- The system should be able to detect and represent graphically the heart beat signal and be able to use diagnostic subsystems for identification of alarming situations. Also it should be possible to expand it to full decision support system able to diagnose minor health situations and provide the information to the doctor. The information should be represented in structured form for further use for data analysis, knowledge base formation and data mining.
- Another objective in the Healthcare scenario is to enable a NoC firewall to protect a hospital gateway from a malicious process. For this scenario, the server must secure critical data from malicious attacks, and the NoC Firewall mechanism can be configured to protect sensitive data and shield server applications (e.g. drivers, diagnosis subsystem and visualization software) from unauthorized access to physical memory regions containing critical data. More specifically, by configuring a set of deny rules via the NoC Firewall driver, one can provide protection of the physical memory assigned to a user-space application from attacks originating from malicious user-level applications, malicious modules (e.g. drivers), and corrupt/malicious devices.

In respect to the healthcare scenario, we first implement a healthcare subsystem that should be able to trace and represent graphically the heart beat signal (electrocardiogram, or ECG) from the Body Gateway device. Data obtained will be further used by an appropriately-calibrated diagnostic subsystem for identifying alarming situations and taking further action. The decision support system must be able to diagnose health issues (e.g. arrhythmias) and annotate useful healthcare diagnosis data on the ECG signal for the physician. Within this process, healthcare data is represented in a structured form not only for data analysis, but also for future knowledge-base formation and data mining which fall beyond our objectives.

In relation to healthcare security, we focus on protection of the Body Gateway driver using two different NoC firewall devices implemented on Zynq 7Z020 FPGA. In the first scenario, we show how multi-compartment isolation can be used to thwart threats from a malicious (or corrupt) kernel-level attack to the Body Gateway driver process. In this case, we adapt a high-level security service (event monitoring, visualization, alert functions) built on top of an existing NoC firewall device (developed by TEI and ST within FP7/TRESCCA), implementing and examining performance overheads. In the second scenario, we design and implement an extended NoC Firewall which is attached to each port of a router, whereas firewall deny rules depend not only on the physical address, but also on the input and output ports of the router that the memory request from the processor is routed through. This new prototype allows experimenting with process-aware group-based key management for supporting privacy of patient data.

Future research related to on-chip security could focus on effectively synchronizing NoC Firewalls distributed across the MPSoC during dynamic updates of rules (addition, deletions, and modifications). During this time it is necessary to guarantee that pending transactions across all NoC Firewalls are processed by the same set of rules. One possible way to do this is via a central barrier operation on the processor; in addition, it is necessary to enable the rule update process to control the AXI protocol handshake, thereby blocking outgoing traffic while rules are being updated at each NoC Firewall. A solution can be based on completion-wait principles.

4 Technological results

Each DREAMS technological result that is present in the Healthcare demonstrator is documented in this chapter. The state, integration with the final target, as well as a number of preliminary metrics on their performance for each result can be overviewed in the next sections.

The following list of technological results, together with the hardware platforms and the overall use case description, reflects the current state of the demonstrator and what has been achieved so far in terms of integration and functionality. This serves as a preliminary assessment, and while not exhaustive of what is expected for the final demonstrator, it enables a first view of what is anticipated from the final assessment report in D8.3.2.

As such the list of technological results covered in the deliverable are related to:

- Tooling/modeling and the virtual platform
- Bandwidth regulation and scheduling policies
- The secure monitor firmware layer
- Network communication with TTEthernet

4.1 Model-Based Development and Tooling

In this section, we describe the foreseen application of the DREAMS development process (D1.3.1 [4]) and tool chain (D4.4.1 [5]) in the health care demonstrator, as envisioned at the current state of the project. The actual application and outcomes will be described in the final assessment report D8.3.2 [6].

We recall that the DREAMS meta-model (D1.4.1 [7] and D1.6.1 [8]) for the description of mixed criticality systems is the backbone of the model based development process and the tool chain. Throughout the development process, the DREAMS meta-model based system description is progressively enriched and verified and finally used to generate the configuration files of the different platform building blocks. The following categories of tools are available for supporting this process:

- modelling
- design
- verification

- configuration file generation

The model editor is used to describe the applications, the technical architecture and the constraints. Then design tools are used, where possible, to automatically create incrementally different parts of the system configuration, which are manually completed, where necessary, with the help of the model editor. Verification tools allow checking the correctness of the automatically or manually created configurations. If a design tool produces a (sub-)configuration that is correct by construction, then the role of the verification tools is that of cross checking and of checking the combination of the different sub-configurations. Finally, the configuration file generators allow translating automatically the verified system configuration into platform configuration files, without errors that would be introduced by "manual" translation.

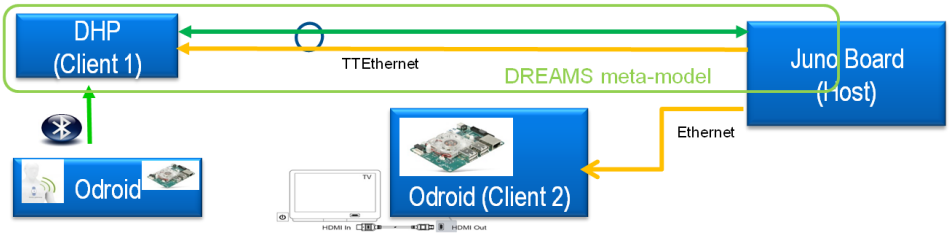

Figure 9: Demonstrator parts covered by the DREAMS meta-model

Given the structure of the Health Care demonstrator D8.1.1 [9], the technical choices described in this deliverable, the available design tools (D4.1.3 [10]) and configuration file generators (D4.2.2 [11]), the usage of the tools listed in the table below is envisioned. In **Error! Reference source not found.**6 the corresponding DREAMS model coverage of the demonstrator is depicted. Regarding the development process, the "Timing Approach" (D1.3.1 [4]) is relevant.

Category	Tool	Purpose
modelling	Autofocus3 (AF3)	Manual creation of the following model items: • Logical Architecture: applications with their communication and timing properties and constraints ○ Healthcare monitoring ○ Statistics ○ Healthcare Application ○ Video Server ○ Free to air • Technical Architecture: • Client1 (DHP) with tiles, on-chip network, off-chip gateway • Host with off-chip gateway • off-chip network, connecting Client1 and the Host Other model items are created with the help of design tools: • On-chip and off-chip communication schedules Aspects that are not covered by the design tools or necessary adaptation are defined with the model editor.
design	RTaW-Timing/ On-chip-COM	Generation of transmission phases for the communication of time-triggered VLs over the on-chip network.
design	TTE-Plan	Generation of communication schedules for the transmission of time-triggered VLs over the off-chip network.
verification	RTaW-Timing/	Evaluation of delays and verification of timing constraints.

	Evaluation	
configuration	AF3/ Conf-File-Gen/ Xtratum	Generation of configuration files for the Xtratum hypervisor.
configuration	AF3/ Conf-File-Gen/ On-chip NI	Generation of configuration files for the on-chip NI in Client1.
configuration	TTE-Plan	Generation of configuration files for the off-chip network.

4.2 Virtual Platform:

Based on available specifications, we have modeled STNoC backbone technology as accurately as possible by making several adjustments to the 5-stages Garnet fixed pipeline model [14], [16], [17]. Several STNoC configuration parameters have been modeled, including link width, packet flits (header and body), virtual circuits (high and low priority), buffer size and number of credits per virtual circuit (VC) and router and NI port. We have also implemented STNoC QoS policies, such as memory interleaving and fair bandwidth allocation for rate control. The latter policies apply to flpol travelling on the same VC. More specifically, our gem5 STNoC router configuration supports three levels of arbitration based on info available in the header flit of the STNoC packet: 1) current faction bit used as a an epoch, i.e., separating messages injected to a router; 2) packet priority, round robin or least recently used (LRU); 3) round robin or LRU as third level of arbitration (this is only used when packet priority is the second level).

In relation to the STNoC router model, we have encapsulated garnet switch allocation (port scheduling) within the VC allocator, while also reducing the pipeline depth to match STNoC specifications. Although in our current setup we assume an STNoC topology (normal spidergon) of degree 4, many different topologies with a maximum degree 5 can be modeled by modifying python configuration files. The current implementation of gem5 STNoC router model uses the internal Garnet routing tables. This allows not only to support the only commercially-used STNoC routing strategy, i.e., source-based scheme, but also other deterministic, randomized or adaptive policies for design space exploration. In our gem5 STNoC model, router-to-router, NI-to-router and router-to-NI LT takes no cycles, similar to the actual STNoC synchronous link implementation. NI latency takes one cycle, which corresponds to STNoC flit registering. The NoC clock frequency can also be configured appropriately (default value is 10^9 ticks per second).

Using our gem5 STNoC, we plan to evaluate important technologies related to the demonstrator, by a) investigating scalability and power/performance tradeoffs of memory interleaving support during DMA operations in NoC-based multicore SoC and b) evaluating QoS-security tradeoffs when protecting privacy of patient data from malicious processes using high-level network security solutions built on top of low-level drivers of a hardware-based firewall module that enables process-centric, path-based memory protection.

4.3 Bandwidth regulation policies at Linux kernel and user-level

Scheduling has been considered in several contexts, including at the instruction level, in network packet switching, and in computation engine processing (CPU bandwidth). Depending on the context's characteristics, different sets of aims and purposes are seen as more important and easier to attain, for scheduling. Usually, a scheduling hierarchy is constructed at the system level, and aims that may not be pursued at fine granularity are handled at a coarser granularity.

Scheduling operates over a set of entities (work items) that it manages by allocating resources to them. These entities may be, for example, program instructions, application tasks, or network packets, depending on the granularity and the context. In addition, schedulers are usually exposed also to higher-level entities (called work contexts) that "own" the scheduled entities, and which compete for the utilization of resources. Thus, instructions and tasks belong to computation threads, and packets belong to network flows.

MemGuard access control is focused on per-core allocation of the minimum guaranteed memory bandwidth (denoted r_min in the algorithm), i.e. the bandwidth that can be guaranteed even for the worst-case memory access patterns. This metric intends to capture the effects of worst-case DRAM traffic patterns, which consist of repeated accesses of the same memory bank, on different bank rows each. It is essential to note here that guaranteed bandwidth (r_min) is significantly less than the maximum attainable memory bandwidth (e.g., usually close to 20%), thus, it is important to favour, as much as possible, best effort traffic (BE), i.e., traffic in excess of r_min.

Within WP2 (deliverables D2.3.3) TEI considers the implementation and evaluation of an efficient Linux kernel module (called Extended MemGuard) for bandwidth regulation on ARM v7 (and in the near future ARM v8 architecture). MemGuard module can be used for differentiating between rate-constrained and best effort messages. Our extension supports a violation free operating mode for rate-constrained flows, and provides dynamic adaptivity through EWMA prediction.

Ongoing TEI integration effort towards the final demonstrator (WP2 deliverables D2.4.1, and D2.4.2) will focus on configuration/parameterization of MemGuard for handling video streaming (part of the DREAMS WP8 Healthcare Demonstrator), most likely through X11 forwarding. This effort will lead to different future implementations of MemGuard, either within the Linux scheduler (as an extension to SCHED_NORMAL policy) that manages mixed criticality traffic by different cores, processes or VMs, or as a new alternative to Unix system-level tools that performs network bandwidth regulation.

4.4 Scheduling heuristics for KVM

As described in D2.2.1 [12] and D2.3.2 [13], for the purpose of the healthcare demonstrator, different scheduling enhancements have been implemented. So far storage I/O and task scheduling extensions have been developed and integrated in the demonstrator, essentially targeting the hospital gateway (Juno platform) and the KVM hypervisor. At a later stage Memory bandwidth regulation on guests is also planned.

The main idea is that the host scheduler is aware of guest prioritization. This can be achieved by enabling the guest to communicate with the host and inform dynamically when it needs to be prioritized. Figure 7, below, describes the principle of co-scheduling. Each time the guest OS, thinks that its priority needs to be changed or when it executes a real-time program, it sends a request to the host which will modify the current scheduling policy.

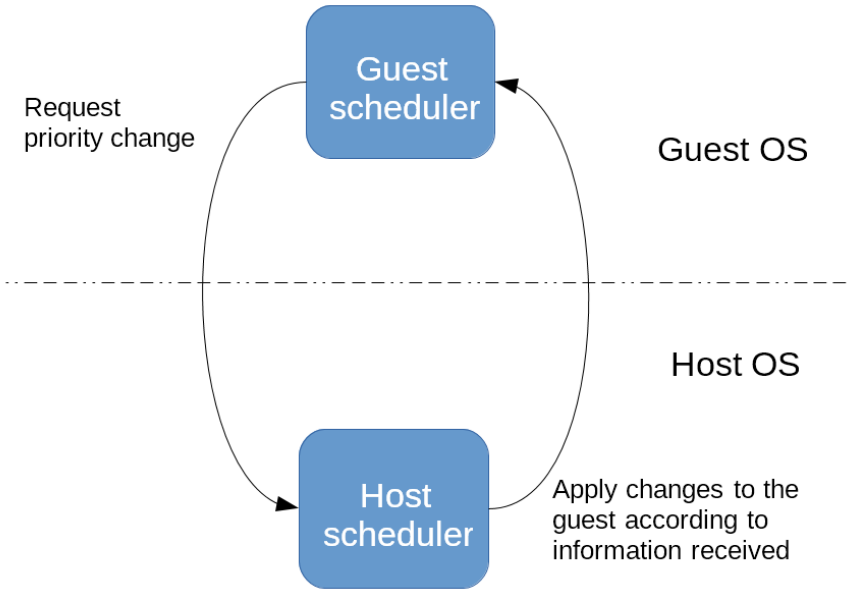

Figure 7: Coordinated scheduling overview

4.4.1 Guest scheduling problem

In the case of storage I/O and task scheduling, any heuristics implemented in the guest are transparent to the host, failing to affect the overall scheduling of the guest by the host. This can be seen with a simple example:

Consider a system running a guest operating system, say guest G, in a virtual machine. Application A, is being started (loaded) in guest G while other applications are already executing without interruption in the same guest. Such a system is represented in Figure 8. In such conditions, the scheduling patterns of guest G, as seen from the host side, may exhibit no special property that allows the scheduler in the host to realize that an application is being loaded in the guest. Hence, the scheduler in the host may have no reason privileging the requests coming from guest G. In the end, if also other guests or applications of any other kind, are executed in the host then guest G may receive *no help* to be prioritized.

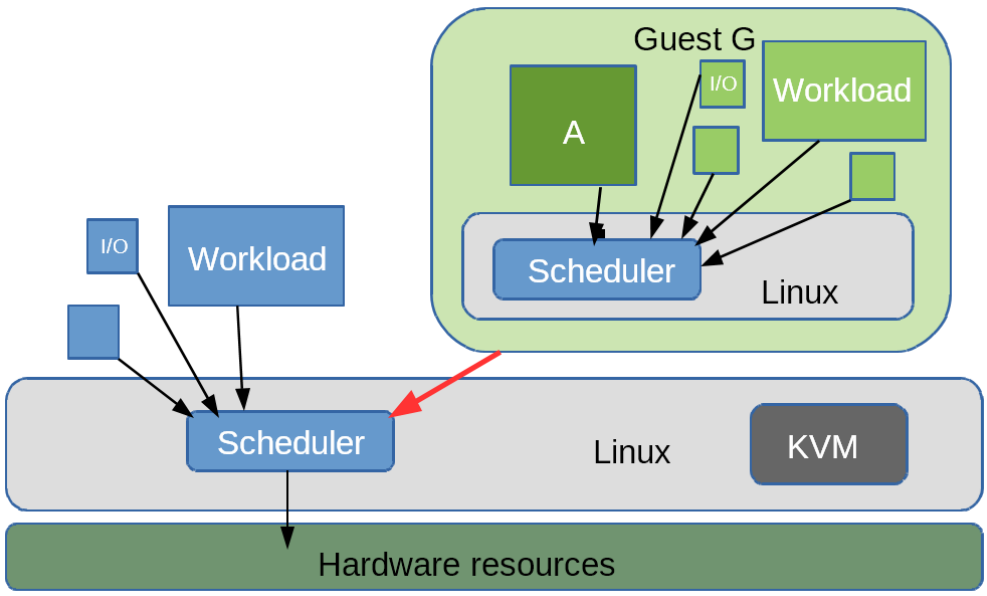

Figure 8: Example highlighting the missing connection between schedulers in virtualized environments

4.4.2 Implementation

For the implementation of the co-scheduling mechanism on ARMv7/v8 architectures, a communication method is required between the host and the guest. For ARM, this can be achieved with the HVC instruction which is considered a hypercall. The Hypervisor Call instruction (HVC) can have an argument that can be used to pass different types of information, for example, a request for issuing to the host a prioritizing period for the guest. With this argument different requests can be defined that will be handled differently by the host.

The co-scheduling mechanism has been implemented on the Linux kernel for the host (use of KVM on ARM) but also for the guest. These modifications imply to modify the kernel code for both guest and host schedulers. On the guest side, the scheduler has to be modified in order to extend any heuristics or scheduling policies with hypercalls in mind. This was done by utilizing paravirt-ops (pv-ops), the hypervisor-agnostic Linux interface.

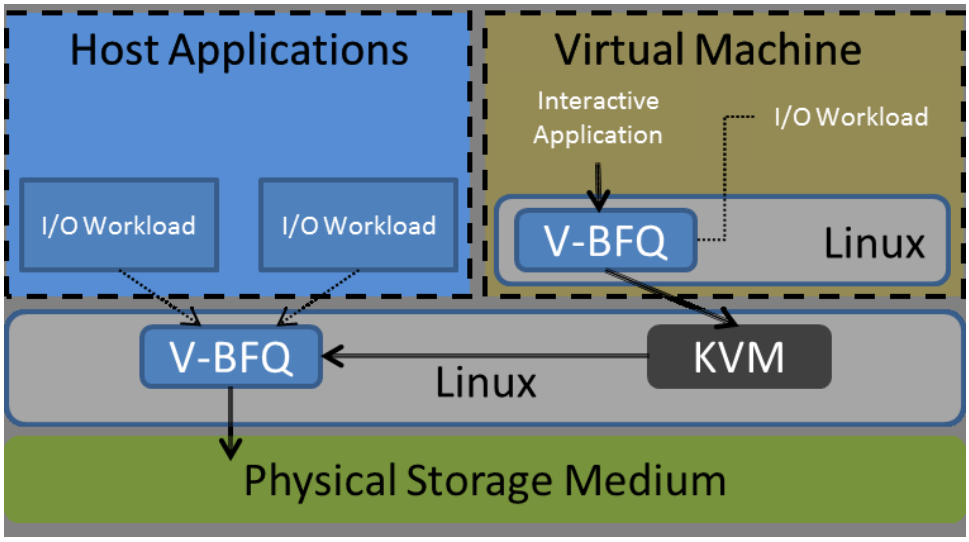

Figure 10: Overview of coordinated I/O scheduling

On the host side, the KVM code has to be modified to handle correctly the HVC calls coming from the guest, additionally the code of the host scheduler has to be adapted to apply with the requests received from the guest. The two schedulers currently supported are BFQ for storage I/O and CFS for the usual task scheduling. Figure 9 depicts the overall architecture of a host/guest system with coordinated scheduling applied on the BFQ I/O scheduler.

4.4.3 Performance metrics

At their current state these scheduling enhancements have shown significant improvement in latency and overall application responsiveness for virtual machines. Although their implementation is not yet finalized or fully integrated with the healthcare demonstrator, preliminary results from the development stage are useful for an initial assessment.

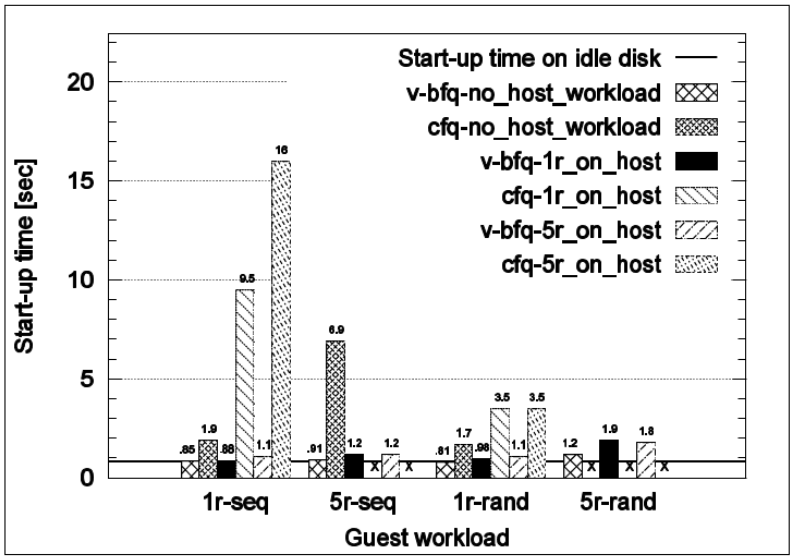

Figure 11: CFQ vs V-BFQ application start-up latency results

Overall I/O guest scheduling is improved for applications that are interactive in nature, making them more responsive. Start-up application latency in synthetic benchmarks dropped to host idle levels, while aggressive I/O workloads were also present. In contrast the default scheduling solution for the guest needs extreme amounts of time to finish the same test (or even fail), as seen in Figure 10. Additionally in real scenarios, video playback was improved with no or less stuttering artifacts compared to defaults.

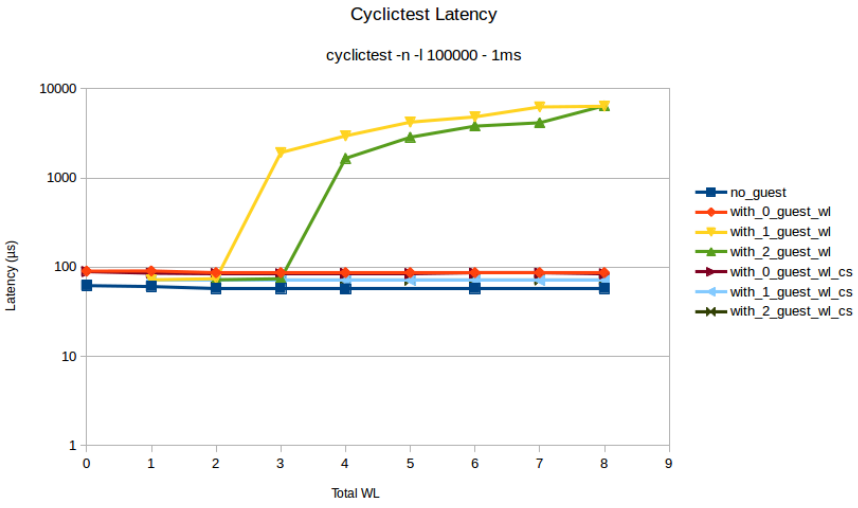

Figure 12: Interrupt latency results with CFS and co-scheduling mechanism

For task scheduling and interrupt latency, the improvement is also significant, where with coordination the guest latency is dropped nearly to host levels (virtualization overhead still present). Coordinated scheduling can dynamically change the priority of the guest when needed, minimizing unnecessary context switches during the execution for a critical task in the guest. This can be seen in Figure 11, where the interrupt latency with Cyclictest is ranging from 2000 to 9000µS in default situations, while with scheduling enhancements is kept at a steady 100µS.

4.5 Secure monitor firmware layer

In order to be able to execute mixed-criticality workloads and properly guarantee hard and soft real-time latency, a secure monitor firmware layer has been implemented specifically for the needs of

the Healthcare demonstrator and the DREAMS project. This firmware essentially allows the concurrent execution of two different operating systems, ensuring their temporal and spatial isolation by means of hardware and software support.

The secure monitor firmware implementation is based on the TrustZone security extensions, which is supported by most modern ARMv7 and ARMv8 processors. TrustZone implements in hardware the concept of different execution modes, called the Secure and Non-secure world. Additionally, properly supported resources can be partitioned to Secure and Non-secure, as for example, memory, peripherals, interrupts and even timers. Secure world protection is ensured by monitoring physical access to memory or peripherals, therefore, a trusted OS, running in Secure world, is totally isolated from applications executing in the Normal world.

The role of the secure firmware is to properly initialize the system and divide resources, as well as to manage the context switching between the two worlds by triggering a Secure Monitor Call (SMC) instruction or by hardware exception mechanisms, such as interrupts (e.g., FIQ, IRQ, External abort). It also oversees these exceptions in order to ensure a correct operation for each world.

The secure monitor firmware has been designed to meet the following requirements:

- RTOS (critical applications) and GPOS (Linux/KVM) co-execution on the processor
- Isolation of RTOS resources (Memory, Peripherals, etc) from GPOS illegal access
- Minimal boot time overhead for the critical RTOS, which should be less than 1% of the original
- Minimize the latency impact - Context switching time must be lower than 1µs
- Firmware footprint must be as compact as possible to take into account certification
- Target architecture is ARMv8-A with security and virtualization extensions

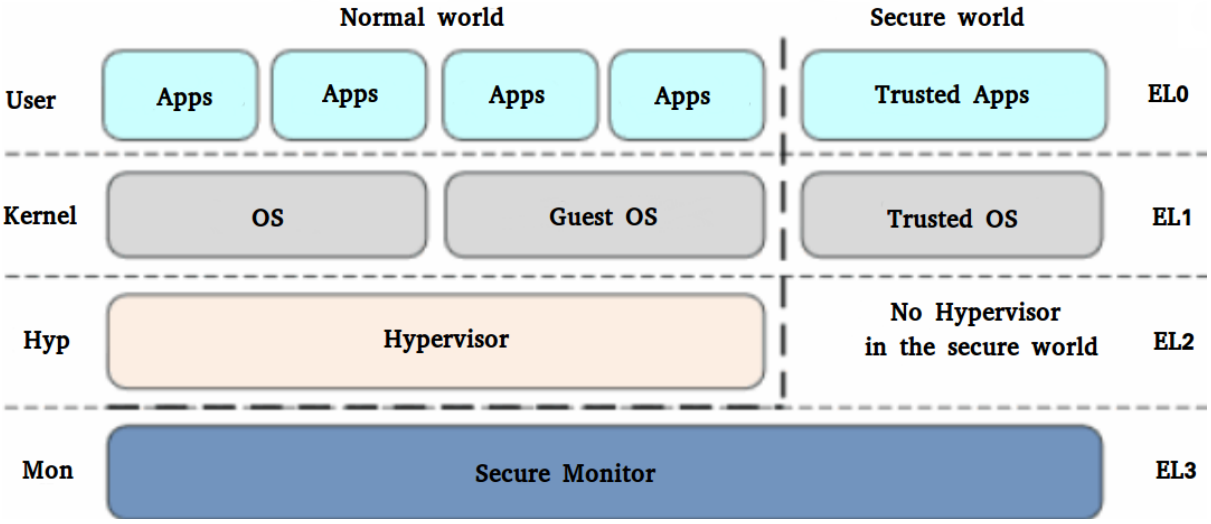

Figure 13: Overall system divided by the Secure Monitor firmware in two different worlds

Figure 12 depicts the two different worlds and their exception levels (EL) – also known as execution modes, as well as the secure firmware, which lies in the most privileged EL and responsible for the monitoring and operation of the whole system. For the needs of DREAMS and the Healthcare demonstrator, FreeRTOS is selected as the trusted OS in the Secure world, while Linux and KVM virtual machines are executed in the Non-secure world. This particular setup combines the advantage of secure isolation for the RTOS while at the same time provides a feature rich environment with Linux and virtualization features with KVM.

At its current development stage the Secure Monitor is able to boot FreeRTOS and Linux/KVM with most basic functionalities of the Operating Systems supported. For the time being, Power State Coordination Interface (PSCI) capabilities are not supported, so SMP functionality is not yet implemented. Both FreeRTOS and Linux/KVM share the same core and the context switch procedure

is handled by the Secure Monitor. Finally the Memory Management Unit (MMU) is not yet utilized, and by consequence L1/L2 caches are also disabled.

4.5.1 Performance metrics

The current performance assessment of the Secure Monitor firmware implementation includes measurements regarding boot-up time, context switch latency of the executed operating systems, as well as general purpose Linux benchmarks like cyclictest and hackbench, to measure the overhead of the overall system configuration. The latency performance is expected to be higher than the target, although this will be alleviated once in the next version of the firmware, MMU/cache support is added.

Measurements are performed by means of microbenchmarks by utilizing the ARM Performance Monitoring Unit (PMU). The PMU, among other things, allows to have a clock-cycle granularity when measuring the time needed for specific functions to be completed. All measurements where performed with one of the two Cortex-A53 cores present in Juno, clocked at 700MHz, which roughly translates to 1,429 ns per clock cycle.

The first type of measurement is the total time needed for the Secure Monitor firmware to configure and boot the system. This interval is defined as the entry point of the Secure Monitor up to the point just before entering FreeRTOS. The total time needed for booting the system is on average around 23 µs or 16000 clock cycles.

The context switch between the Secure (FreeRTOS) and the Non-secure (Linux/KVM) world is defined as the total amount of time needed to pass the execution from one world to another. This measurement is also related to the worst case interrupt latency for the Secure world, meaning that a secure interrupt fired while the normal world was scheduled. For this scenario preliminary results show on average a latency of 11 µs or close to 8000 clock cycles.

Linux standalone on 1 core		
Hackbench (s)	Cyclictest (µs)	
	Avg	Max
3,103	**12**	**117**
Linux/FreeRTOS (low) co-execution on 1 core		
Hackbench (s)	Cyclictest (µs)	
	Avg	Max
3,205	**14**	**139**
Linux/FreeRTOS (high) co-execution on 1 core		
Hackbench (s)	Cyclictest (µs)	
	Avg	Max
11,272	**831**	**2172,6**

Table 1: Linux benchmark results with different co-executed workloads

On the Linux side, the cyclictest and hackbench benchmarks are used to estimate the performance and latency overhead when running in parallel FreeRTOS. Results were compared to default values where Linux is the only software executed. For cyclictest the reported latency can change depending on the amount of CPU load existing on FreeRTOS (which is prioritized by default). In low FreeRTOS workload conditions, the maximum latency increased from the default of 117 µs to 139 µs. In the case of overcommitting the system the latency overhead toped at 2172 µs, which is expected since FreeRTOS will starve out Linux from resources if too many tasks need to be scheduled without any interruption. Figure 13 sums up the results for the Linux performance and latency benchmarks.

4.6 Off-chip network

The on-chip/off-chip network interface is integrated in the hardware platform that is used for the demonstrator.

Figure 14: PCIe TTEthernet device by TTTech

The off-chip gateway is integrated on the TTEPCIe device by TTTech (Figure 14) which consists of a TTEXMC Card and a passive PCIe COTS carrier board. It can be used in PCI Express (PCIe) x4, x8, and x16 slots and supports three SFP channels which, in the demonstrator, are connected by means of standard RJ45 cables. In the context of WP8, the gateway services are ported to the Altera Stratix IV FPGA that is available on the TTEXMC Card providing the DREAMS-specific services in hardware. The TTEthernet device is connected to the ARM Juno Board by PCI express in order to provide the gateway services to the applications as described in chapter 2.

The off-chip gateway is furthermore integrated in the DREAMS Harmonized Platform (based on the Xilinx Zynq-7000 SoC) which is used in the scenario, as depicted in Figure 1. Both devices, i.e. the Juno-board and the DHP, communicate with each other using the real-time communication services offered by TTEthernet, based on time-triggered and rate-constrained messages.

Figure 15: TTEthernet 24 port switch by TTTech

To this end, they are connected to each other by means of a TTEthernet switch provided by TTTech. In addition to the two end-systems depicted in the figures above, also non-critical devices and services can be attached to the switch with the guarantee of non-interference in the communication. The switch hosts DREAMS-specific firmware in order to provide the modified communication services (in particular: security services). The device is depicted in Figure 15.

4.6.1 Off-chip network communication

The healthcare demonstrator utilizes the mixed-criticality network in order to communicate between the different nodes that are used, in particular between the DHP and the Juno board (both

critical and non-critical traffic) and the ODROID device (only non-critical traffic) as depicted in figure 6 .

For the critical traffic, time-triggered and rate-constrained communication links are required. In order to support these links, the integration of the communication stack in the hypervisors running in both systems was required. Different hypervisors are used on these devices, i.e. XtratuM for the DHP and KVM for the Juno board.

4.6.2 Configuration tools

In order to configure the three devices that utilize the communication services, the two tools TTE-Plan and TTE-Build are used as depicted

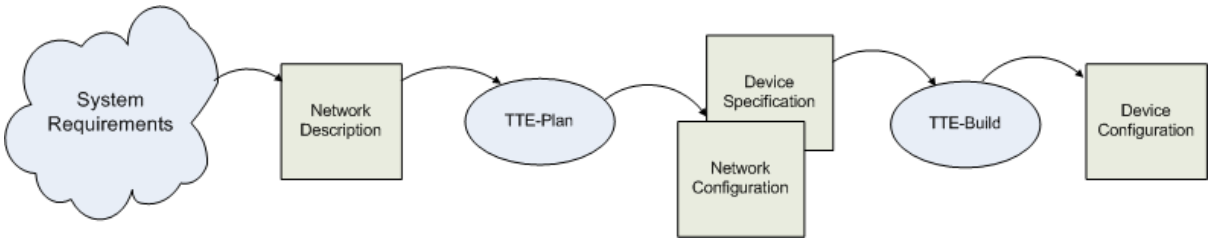

Figure 16: TTEthernet tools for network configuration

A detailed description of how they are used to create the necessary configuration files is provided in deliverable D4.2.1 (chapter 4.2 TTEthernet Network).

5 Evaluation Methodology

In this chapter, the methodology to evaluate the project approach on the basis of the Healthcare demonstrator is defined. A preliminary 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 [14] Part B, section 1.1.

5.1 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 measure 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, estimation will be provided in the final report.

Table 2 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	Time
1	Achievable Safety Integrity Level	Maximum achievable Safety Integrity Level (e.g. ASIL-B, ASIL-C) according to ISO 26262 [15] [16] [17] 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
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
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 Realiability	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
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 Real-time constraint for: a) 24 FPS, half-screen size, 24-bits/pixel, peak=90%,, avg=85% (A57 cluster) b) 20 FPS, half-screen size, 16-bits/pixel	1.4	E
33	Juno R1 memory 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 Real-time constraint for: 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 Real-time constraint for: 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. Real-time constraint for: 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 2: Key Performance Indicators

Table 3 collects the values of the KPIs that can be evaluated at this stage of the project (i.e., KPIs tagged with 'D' in Table 2). 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	No ASIL	The secure monitor firmware layer has been designed to meet the stringent requirements of the ISO 26262 certification. The ASIL-C certification of this software component is planned for 2017 - H1.
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. Evidences of this performance measurement can be extracted from D2.3.2 [18].
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%. Evidences of this performance measurement can be extracted from D2.3.2 [18].
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 [18]. 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 [12], 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 [12].
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 [13]. 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 [13]. The implementation isolates each virtual machine regarding the executed task.

Table 3: KPIs evaluated at M32

5.2 Objectives assessment

The following tables present the progress towards the completion of measure for success and project objectives by analyzing available information at this stage of the project. The measures for success are marked with green color 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 of the secure monitor firmware layer is planned for 2017-H1. Regarding the ST Bodygateway, more tests will be performed in the final assessment report.	
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. Therefore, real-time objectives will be achieved.	
1.3 Fault containment	10	This measure for success cannot be evaluated yet, since the corresponding KPI is not yet assessed	
1.4 Timely adaptation	32, 33, 34, 35	This measure for success cannot be evaluated yet, since the corresponding KPIs are not yet assessed	
1.5 Security			
1.6 Domain-independent core services			
1.7 System Modelling (i.e., fine	11, 12	This measure for success cannot be evaluated yet,	

grained analysis / scheduling, complexity, completeness)		since the corresponding KPIs are not yet assessed.	
Objective evaluation			
Although most of the measures for success cannot be evaluated, available data suggests a positive progress.			

Table 4 : 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	Memory bandwidth isolation is guaranteed by the memguard-kvm implementation, whereas critical applications are isolated through the secure monitor firmware relying on ARM TrustZone. More tests will be performed in the final assessment report in order to evaluate the impact of non-critical application on critical one.	
2.2 Reduced bank conflicts	17	This measure for success cannot be evaluated yet, since the corresponding KPI is not yet assessed.	
2.3 Gateways	13, 14	This measure for success cannot be evaluated yet, since the corresponding KPIs are not yet assessed.	
2.4 Reduction of latencies	6	The co-scheduling implementation for KVM virtual machines allows minimizing the overhead.	
2.5 Reduction of jitter	18	This measure for success cannot be evaluated yet, since the corresponding KPI is not yet assessed.	
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			
The preliminary evaluation of this objective is positive, but there is some information missing which will be covered in the final assessment report.			

Table 5: 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. However, more tests will be performed in the final assessment report in order to evaluate the impact of non-critical application on critical one.	
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	This measure for success cannot be evaluated yet,	

		since the corresponding KPI is not yet assessed.	
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			
The preliminary evaluation of this objective is positive, but there is some information missing which will be covered in the final assessment report.			

Table 6: 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	This measure for success cannot be evaluated yet, since the corresponding KPI is not yet assessed.	
4.3 Automatically executable transformations	20	This measure for success cannot be evaluated yet, since the corresponding KPI is not yet assessed.	
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is no information at this point.			

Table 7: 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		This measure for success cannot be evaluated yet.	
5.3 Architectural support		This measure for success cannot be evaluated yet.	
5.4 Configuration optimization			
5.5 Variability		Critical applications (e.g., bandwidth, peripheral, memory, etc) are isolated from faults which occur in other partitions. However, more tests will be performed in the final assessment report in order to evaluate the impact of non-critical application on critical one.	
5.6 Domains and market features	21	This measure for success cannot be evaluated yet, since the corresponding KPI is not yet assessed.	
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is no information at this point.			

Table 8: 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	Although the secure monitor firmware layer has been designed to meet the stringent requirements of ISO 26262, it is not yet certified.	

6.3 Cost		This measure for success cannot be evaluated yet.	
6.4 Reusability		This measure for success cannot be evaluated yet.	
6.5 Extensibility		This measure for success cannot be evaluated yet.	
Objective evaluation			
Although most of the measures for success cannot be evaluated, available data suggests a positive progress.			

Table 9: Objective 6 assessment

Objective 7: Promoting widespread adoption and community building			
Measure for success	KPIs	Evaluation	
7.1 Community infrastructure		This measure for success cannot be evaluated yet.	
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		This measure for success cannot be evaluated yet.	
7.4 Roadmap		This measure for success cannot be evaluated yet.	
Objective evaluation			
Although most of the measures for success cannot be evaluated, available data suggests a positive progress.			

Table 10: Objective 7 assessment

6 Conclusion

In this document the current status of the Healthcare demonstrator is reported, including the state of the hardware platform the planned use case and scenarios, as well as the first set of technological results that are currently integrated. Related to the hardware arrangement, the main platforms that are so far intermediately integrated are the DREAMS harmonized platform and Juno development board targeting the ARMv7-A and ARMv8-A architectures respectively.

On the software side, the main components used are the XtratuM hypervisor with a variety of partitions on the DHP. For the Juno development platform and with the infrastructure provided by the Secure Monitor firmware, FreeRTOS is used as real time operating system which is executed concurrently (and securely isolated) with Linux (GPOS). Alongside Linux, KVM is used to instantiate additional virtual machines with enhanced scheduling between the guests/host, resulting in lower latency and increased responsiveness by means of coordination. Finally, the body gateway is now successfully used with software that renders the patient's heart beat data.

For the next steps, it is important to extend the definition of the end user software, including the XtratuM partitions, KVM guests and host applications. Additionally, the integration of the TTEthernet with the Juno development platform is not yet realized (drivers have been tested but the PCI device is not yet available), and the connection between DHP and Juno has not been tried yet. Finally, for the next assessment report (D8.3.2), the target is to document the full range of performance metrics on the actual demonstrator and the complete integration of the DREAMS technological building blocks with the use case.

7 Bibliography

[1] DREAMS D1.5.1 Intermediate integration of DREAMS platform with virtual platform prototype, DREAMS Consortium, 04/2015

[2] Xilinx, ZC706 Evaluation Board for the Zynq-7000 XC7Z045 All Programmable SoC - User Guide, http://www.xilinx.com/support/documentation/boards_and_kits/zc706/ug954-zc706-eval-board-xc7z045-ap-soc.pdf

[3] ARM, 64 Bit Juno ARM Development Platform, https://www.arm.com/files/pdf/Juno_ARM_Development_Platform_datasheet.pdf

[4] D1.3.1 - Description of development process with model transformations,» DREAMS Consortium, 7/2014

[5] D4.4.1 - Tools feature map and interoperability capabilities, DREAMS Consortium, 7/2016

[6] D8.3.2 - Assessment report for mixed-criticality healthcare and entertainment use cases, DREAMS Consortium, 2017

[7] D1.4.1 - Meta-models for Application and Platform, DREAMS Consortium, 3/2015

[8] D1.6.1 - Meta-models for platform-specific modelling, DREAMS Consortium, 5/2016

[9] D8.1.1 - Specification of the use cases along with technologies and assessment metrics, DREAMS Consortium, 2015

[10] D4.1.3 - Final implementation and improvement of the offline adaptation strategies for mixed criticality, DREAMS Consortium, 7/2016

[11] D4.2.2 - Final implementation of a platform configuration files generator, DREAMS Consortium, 7/2016

[12] 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

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

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

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

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

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

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