Flexible and timely on-line integration of medical services using iLand middleware

Marisol García-Valls[1], Natividad Herrasti [2], Christophe Jouvray[3], Aintzane Armentia[4]
[1]Universidad Carlos III de Madrid, Spain
[2]Embedded Technologies, Spain
[3]Trialog, France
[4]University of the Basque Country, Spain
[1]mvalls@it.uc3m.es
[2]nherrasti@embedded-technologies.org
[3]christophe.jouvray@trialog.com
[4]aintzane.armentia@ehu.es

ABSTRACT

Modern medical systems are intensive in the use of distribution technology in medical services. Communication among professionals and patients is enabled to exchange patient information, and for on-line monitoring of health conditions. A miriad of specialized middleware solutions that have appeared in this domain, typically focus on providing more and more patient-data processing services over basic communication backbones (e.g., from pure socket communication to more sophisticated interaction models). Paradigms such as service oriented architectures and decoupled communication middleware support the development of emerging medical systems based on the composition of existing (posibly remote) services into a new application. This adjusts to the phylosophy of *systems of systems* (SoS) development. In a cyber-physical medical domain, service based applications require support for this composition in a way that a new application is created (or simply modified) in a timely fashion. iLand middleware supports this phylosophy, enabling both off-line and run-time reconfiguration of distributed service based applications. It provides timely operation, controlled communication delays, and a decoupled interaction model for stateless services. iLand has been used in medical applications for remote patient monitoring and in surveillance domains for remote real-time video transmission. This paper presents the systems that iLand has enabled in the medical domain, mainly based on remote patient monitoring. These iLand-based systems have shown to be flexible and capable of undergoing timely service reconfigurations.

Keywords

Cyber-physical medical systems; middleware; service oriented applications; flexible service integration

1. INTRODUCTION

Communication middleware is essential in current and future medical systems. As a matter of fact, communication middleware technologies are at the heart of almost every single distributed application nowadays. They are characterized by abstracting the low level details of the communication protocols and the hardware characteristics to programmers. This way, they can focus solely on the functional aspects of the application, and increase their productivity and the application maturity level.

The first middleware technologies provided mostly RPC (remote procedure call) interaction schemes. With the years, they evolved towards object-oriented programming such as CORBA [24], or Java RMI [31], among others; and later publish-subscribe data-centric paradigms such as DDS [26], JMS [5], or AMQP [22]. Some of the available middleware technologies provide different communication models such as ZeroC Ice (Internet Communication Engine) that offers both remote method invocations and asynchronous publish-subscribe through its version Ice Storm.

Over the years, middleware technologies were fine tunned to suit the specific needs of individual application domains, such as stock market [23], transport [21], or health care [2]. Middleware technology is constant progress to suit the needs of the emerging domains such as real-time cloud computing [10, 11]. Middleware is important to healthcare systems for a number of reasons; just a few of them are:

- It acts a an interoperability provider, allowing information (patient records) to be communicated/transfered and formated to the specific needs of physicians.
- It provides major flexibility across departments, as they may be interested in different patient data, that can be extracted as required from a global repository.
- Eases evolution to richer functionality in short time such as hospitals and care centers in general, as new functions can be developed as decoupled services.
- It allows the connection of mobile devices to the hospital network cloud, making data available to professionals anytime anywhere.
- Patient engagement in their care could be increased as middleware can display their historic records and current monitored data in an understandable way so that

they can participate to their healing process. The engagement of patients enabled by middleware requires that the middleware provide them with anwers in an efficient, timely, and robust manner.

The last characteristic is studied in this paper. Not all middleware technologies are suitable as communication backbones in health care systems. For instance, in a centralised patient monitoring system that receives and displays the information about patients in their rooms/apartments, a bad decision on the selection of a middleware technology may yield to latencies of over tens of seconds; also, the wrong selection may result in lack of support to flexible on-line service integration. This limits the potential of systems that monitors the health of severely ill/aged patients.

This paper presents the characteristics of the open source middleware technology named iLAND [16], showing its application in two different systems related to health care. For the first application of iLand on remote monitoring of daily activity of patients, reconfiguration of the system may take place continously depending on the patient's . For the application of monitoring health conditions of elderly housing, more strict temporal requirements are placed and iLand is executed over a time-triggered communication protocol.

The paper is structured as follows. Section II provides some related work in the area of medical systems. Section III presents describes iLand middleare, its characteristics, architecture and properties. Section IV presents the application for remote daily monitoring of patients activity. Section V describes the implementation of an elderly monitoring system. Section VI concludes the work discusing the implementations and comparing them.

2. BACKGROUND

One of the most frequent uses of middleware in medical systems is the automatic handling of patient records. Current practice often uses health information systems (HIS) and electronic health record (EHR) in an informal manner with adhoc protocols and interoperability solutions in order to develop clinical systems. Typically, attention in these systems is paid at the pure application level where healthcare enterprise systems are put together to deliver specific clinical solutions. Aspects as security and safety are essential in current solutions, specially when envisioning medical cyber-physical systems as a number of subsystems will coexist and interact through a safety/security framework. Such a framework must provide trustworthy compositional techniques to integrate devices, cooperating autonomously and in a protected mode with health information systems. For this purpose, new paradigms are appearing as MAP (*medical application platforms*) [19] that is a safety- and security-critical real-time computing platform. Other architectural solutions have appeared such as ICE (Integrated Clinical Environment) [3, 30] led by CIMIT Medical Device Plug-and-Play interoperability project and later standardized.

ICE approach defines important elements such as the *Protocol stack for medical device interoperability* [3]. The lower layer of this stack is DPWS in charge of service discovery, interface description, messaging, event propagation, and secure information transmission. On top of this pure web service communication level, a streaming dual channel transmission based on MDPWS is provided. ICE device specific extensions are just on top with a BICEPS layer for Basic integrated clinical environment Protocol Specification.

Below this DPWS there indication of a specific technology, a part from the typical de facto usage of HTTP/TCP or UDP. However, no reference to the basic communication middleware technologies (e.g. DDS [25], ZeroC Ice [32], or iLAND [14,16] middleware for service oriented real-time applications, etc.) are indicated. DPWS is bound to using SOAP protocols with messages in XML, yielding to heavy communication latencies and parsing/unparsing times that may not be suitable to all domains.

Studing alternatives to a pure DPWS backbone is needed as it yields to assessing the timely interaction between remote nodes/devices that is more suitable for the inherent temporal requirements of cyber-physical systems. Timeliness is a critical issue for some subsystems of a larger CPS that must be considered. Delays due to message parsing or heavy XML message transmission may yield to delays in a control loop that can destabilize the control over the physical system. Specific alternative solutions for efficient transmissions must be considered for the health domain based on the specific temporal requirements of applications.

3. ILAND MIDDLEWARE

iLand [14] is an open source middleware that has been applied in industrial prototypes, including medical systems. It follows the classical principles of a layered design; though its architecture is independent of the underlying communication network protocol, the reference implementation [16] uses a DDS backbone. iLand includes a number of enhanced functions to support dynamically reconfigurable applications based on services: light-weight services in the real-time version and web services in the soft and best effort version with QoS guarantees. The basic characteristics are:

- Integration of time-deterministic reconfiguration techniques [4, 7] and service-composition algorithms functions [6, 17] that is integrated in the *Core Functionality Layer* (CFL). CFL components are the *service manager* (SM) to define self contained stateless services; *application manager* (AM) to define the structure of larger applications made of service sets (service graph); and the *composition logic* (CL) containing time-deterministic service composition and reconfiguration algorithms;

- Adaptable to both real-time network protocols (e.g., time-triggered for full real-time schedulability analysis) and Internet protocols based on TCP/IP or UDP/IP. This functionality is contained in the *Communication Backbone and Resource Management Layer* (CBL). This layer enables easy porting to different off-the-shelf middleware backbones such as DDS, ZeroC Ice, Corba/RTCorba, Java RMI, JMS, AMQP, etc, by the adaptation of a single bridge component [9, 28] for synchronous or asynchronous interaction models.

In iLand, a reconfiguration is the process of transitioning from the current state of the system to the target state. A service oriented application (an application or a SOA) is the set of services (each service is S_i), that are currently part of the system and that are active. A SOA is specified as a graph. where each service S_i has a number of possible implementations being $s_{i,k}$ the k^{th} implementation of S_i. A system state is the set of service implementations

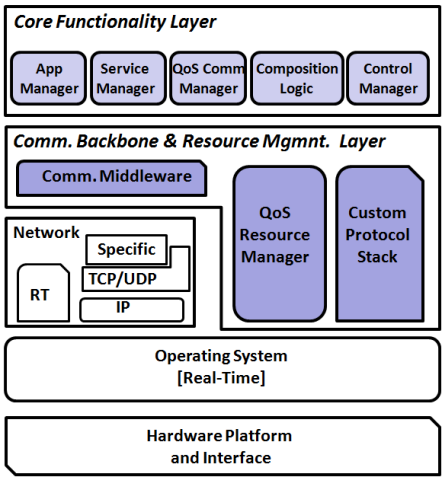

Figure 1: iLand middleware architecture

$(S_{i,j})$ that are active and running. In a reconfiguration, at least one service implementation needs to be either stopped, replaced by another one, or launched. A service implementation $S_{i,j}$ is specified by its functionality, its timing parameters, and the list of dependencies in the following way: $\{F, C, T, D, P, Q, \Delta\}$ where F is the functionality, C is the computation time or processor cycles it requires to complete its function, T is the release period since in our real-time computation model all tasks are approximated by periodic (see [15]), D is the deadline to complete its function, P is the priority that indicates the relative importance of the service implementations, Q is the output quality delivered by S_i, and Δ is the dependency list with respect to other service implementations. iLand reconfiguration logic (reconfiguration event detection, target configuration selection, and transition) is based on the efficient management of the SOA graphs (precisely, of the service implementation graphs) through a set of well defined steps to ensure timely transition [6, 7].

4. REMOTE MONITORING OF PATIENT'S ACTIVITY: A RECONFIGURABLE SERVICE BASED APPROACH

This application intensively utilizes sensors to detect activity in the patients house to both, reconfigure the environment to aid his/her everyday life living and to detect missuse (or similar events) over potentially dangerous electric appliances. A reasoning engine fuses data and actuates on the house environment to both help the patient (possibly partially disabled) and to provide safety features (e.g., gas or oven detectors indicate that the mechanism is on and the patient has left the room or the apartment). Figure 4 shows a prediction mechanism. The system detects the presence and activity of the patient and turns the house environment into a smart area that eases the patient's daily activity.

The patient's house is intensive in the use of sensors and actuators. Sensors monitor the physical systems and the activity performed by the patient. Actuators are devices that change the state of physical spaces or appliances. Moreover, there are a number of different possible interfaces and smart objects for patient interaction (smartphones, tablet, interactive mirrors, etc.). There are two basic types of services monitoring and controlling the sensors and devices that are present in the house: *actuator services* and *interface services*.

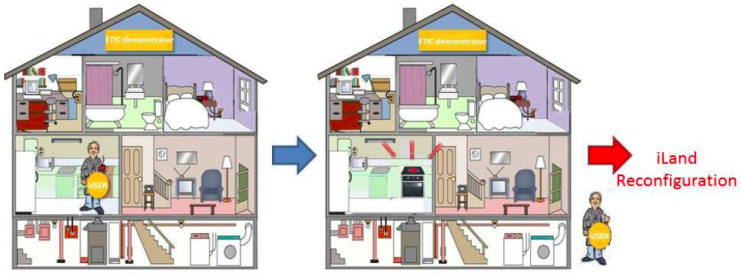

Figure 2: Reconfiguration scenario enabled by iLand in the activity monitoring application

The system has four basic subsystems or parts, each having a set of services:

- *Daily activity detection* that integrates the following services: *Location System*, *Domotic Controller*, and the *Things Manager*.
- *Application logic and reasoning*, containing a *reasoning engine* service that executes the logic to infer the actuation actions.
- *Actuation* that has the *Actuator Service* that is local to each subsystem in the house.
- *Notification* that is performed by the *Interface Service* via opportunistic user interfaces depending on the location of the patient.

Services of each part will make use of different subservices in order to perform their associated activities. This is shown in figure 3.

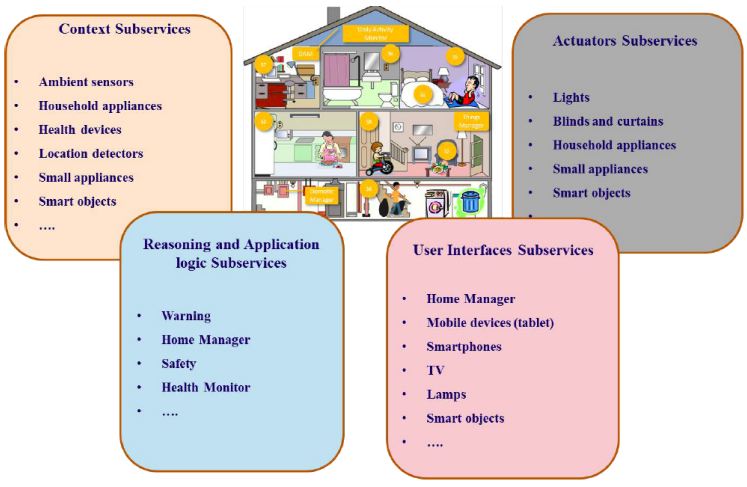

Figure 3: Subsystems and their constituent subservice list

The *daily activity detection* services and *reasoning engine* service are part of the service that is performing continuous monitoring. The *actuator service* and *interface service* are

part of different subsystems that are launched when the reasoning engine triggers a *reconfiguration event*. A reconfiguration event is launched when a modification of the system services has to be undertaken; then, the appropriate service implementations are selected depending on the patient activity and contextual information gathered from the sensors. The reasoning engine decides if it is necessary to launch a new SOA (application graph) or if it is needed to simply replace the current implementation of a given service by a different implementation.

A single service may be implemented in different ways that can yield to different output quality services (e.g., a video display service). There are three main services registered in iLand, that have different possible implementations:

- *Location Service* (LS) that determines the patient location by using different sensors deployed over the house spaces.
- *Domotic Controller* (DC) that monitors and controls the status and operation of different home appliances.
- *Things Manager* (TM) that monitors the use of different household objects, such as: chairs, bed, garbage collector. These provide basic information to reason about the activities performed by the patient. Changes in the standard use of these objects may indicate a change in the patient's health conditions.
- *Reasoning Engine* (RE) that determines the actions to be executed based on the information received from the context services. So, it uses the information from the location service (LS), the domotic controller (DC), and the things manager (TM). The reasoning engine service logic algorithms takes decisions on actuation based on these data. The result may launch a reconfiguration of the service architecture.
- *Interface* (Int) service has the goal of displaying useful information to the user. Examples are, e.g., indications of some appliances being on, or reminders of medicine habits. *Int* will select specific interfaces depending on the patient's location and physical habilities.
- *Actuator* (Act) performs the basic actuation over the devices and house electronic units.

Figure 4 shows the architecture of the application based on services (SOA).

The reconfiguration of the current house service architecture is triggered at specific instants according to the patient daily activities. iLand middleware provides temporal bounds on the duration of the reconfiguration that are expressed in the specification of the SOAs. The reconfiguration protocol of iLand is time bounded and defines a sequence of phases to guarantee timeliness [6,7]. Figure 5 shows the set of steps.

All nodes are running iLand middleware in different configurations. Embedded nodes controlling sensors run a reduced footprint version and the servers (present in a specific location at the patient site); other embedded nodes run the full fledged version with the reconfiguration logic capabilities and the *reasoning engine* (RE) initiates the reconfigurations. The reconfiguration is triggered from the iLand reconfiguration logic (step 1) based on the information gathered from

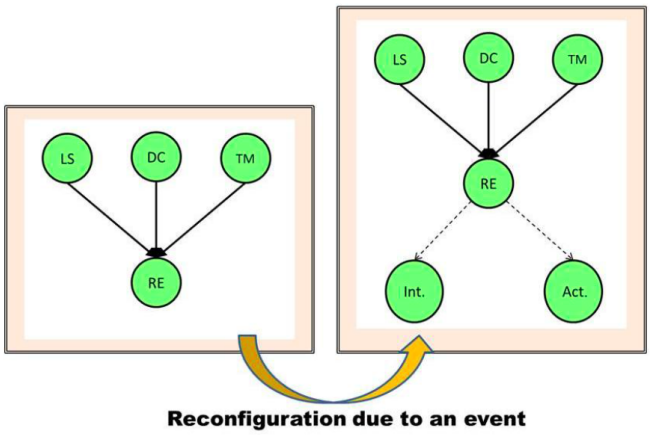

Figure 4: Application services updated after a reconfiguration event

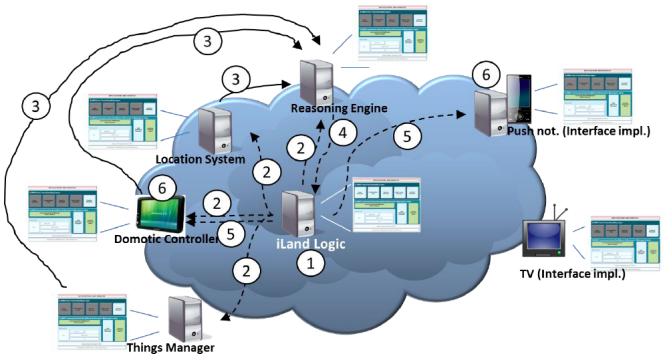

Figure 5: iLand reconfiguration sequence applied to the daily monitoring system

RE (step 2). The reconfiguration actions take place by informing the rest of services, including the *domotic controller* (DC) to either display the right information to the patient or actuate on the household electric appliances and devices.

5. ELDERLY HOUSE MONITORING

This section presents a monitoring system for an elderly house based on iLand middleware. Patients may be in severe physical conditions, therefore they need to be monitored in real-time. They are located in a centralized building (elderly house), each having an individual room specially configured according to the patient's health requirements. Figure 6 presents the general overview of the system. Patient's individual spaces are equiped with specialized medical systems integrated in the room. Patient data are sampled by sensors; then, they are processed, logged, and transmitted to a control center in the building with continuos presence of medical staff.

Each room has a service *room manager* and a set of medical sensor that monitor the patient. The room manager is a front-end monitoring and decision making system that performs basic analysis of the patient. The medical equipment subsystem is simulated, and the gathered data are synthetically generated to test monitoring algorithms with a few physical parameters analysis. The room manager is an em-

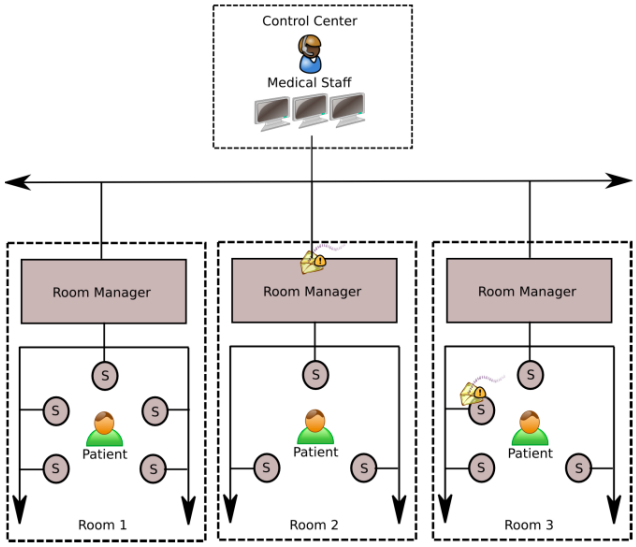

Figure 6: Elderly house real-time monitoring system

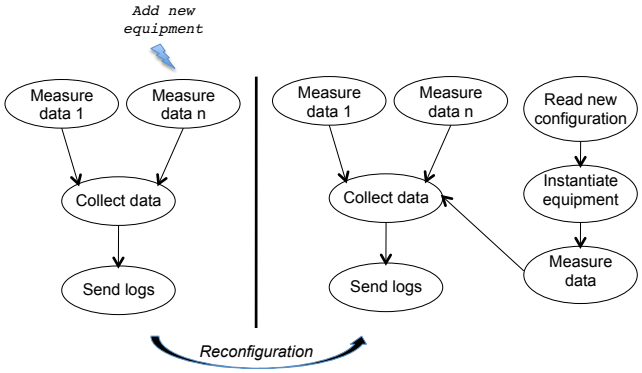

Figure 7: Reconfiguration scenario

Figure 8: Data collection SOA

bedded system with a basic interface, integrated with the room. The control center is equiped with iLand middleware and a data base connection.

The underlying communication protocol has to be highly reliable and timely; data cannot be lost nor suffer unbounded delays. For this system, iLand middleware was adapted to time-triggered communication, that enables real-time schedulability analysis. Time-triggered protocol [20, 27] was analysed to ensure that it provides sufficient flexibility for asynchronous event transmission over Ethernet, like alarms. The usual approach is to fix periodic time slots for alarm events. In this system, in order to reduce bandwidth usage, the protocol has to be adaptive.

Time requirements are specified in this system as there are different criticality levels, some being highly critical. Periodicity of the medical patient monitoring samples range from $1s$ to 1 min, according to the patient health conditions. Data logs to the control center are stored in $1s$ periods. Alarms are sent by a room manager to the control center withing time deadlines of $500ms$ to $1s$ periods.

Diffent reconfiguration scenarios are possible: arrival of a new patient; room adaptation according to patient health conditions; and alarm events as some basic patient health threshold has been reached. Figure 7 presents the SOA (service oriented architecture) for a reconfiguration.

Two SOAs are instantiated for each room: (*i*) *data analysis*; and (*ii*) patient *monitoring/supervision*. *Data analysis* SOA is made of the following services:

- *data collection* (see figure 8) gathers values sampled by the sensors;
- *data analysis*, fuses the data and reasons about the profile of the patient in order to determine the current state of the patient, and commits the result to the local storage;
- *send logs*, formats the sampled values and the state value to be sent to the control center;
- *collect logs* retrieves the logs sent by the room manager;
- *store database* for persistent data storage of monitored information and patient's data.

The list of services of the supervision SOA (see figure 9) are: (*i*) *read database* to retrieve stored logs from the data base; (*ii*) *supervision* that analyses the logs (patient data) on the control center to determine the patient health conditions; and (*iii*) *local supervision* is run by the room manager and performs on-site local results analysis.

An additional SOA performs *log data display* to medical staff in the control center. The log data display SOA has two services: *read DB* (for interfacing to the data base) and *display* for handling data formatting and display.

This design allows to handle each room individually while preserving continuous operation through a single end point for control. As an example, upon arrival of a new patient, it is possible to instantiate the two SOAs needed for data collection and supervision/monitoring in this room to configure them without affecting the rest of the system. Likewise, reconfiguration only concerns the data collection SOA of a single room and not the whole system. Figure 9 shows the relation between the mentioned SOAs.

iLand middleware was ported to run a time-triggered protocol with scheduled time-slots that guarantees real-time asynchronous traffic transmission such as alarms. Measurements over the actual system have been taken to show the stable reconfiguration times provided by the middleware. Figure 10 present the measurements.

Measurements correspond to different experiment with varying elementary cicles (EC) [20] for the time-triggered communication. Stability is evidenced in the reconfiguration times, showing efficiency and real-time transmission deadline preservance for all cases.

6. CONCLUSION

Two different applications of iLand middleware in the domain of medical systems have been presented. Both target cyber-physical medical areas as patients data is monitored (from physical signals to activity indications) in order to reason and react on the current situations, depending of inferred needs. The interaction paradigm of iLand is a de-

coupled one, therefore supporting decoupled integration of services that can be either local in the same node or distributed among different nodes and even in different network segments. iLand middleware has a modular architecture that supports easy porting to different underlying communication middleware technologies, ranging from the most conventional and de facto standards such as DDS or lower level networking protocols operating at the medium access layer such as time-triggered ones. The reconfiguration logic of iLand is based on the one side on graph algorithms that and search techniques that target timely decision making; and on the other side it is based on a set of steps to guarantee the reconfiguration of a whole system based on services. Reconfigurations are supported in a timely way: the modification of a service (replacing a specific implementation) or a SOA (replacement of the service graph) is achieved within the specified deadline, as show the time measurements performed to the elderly house case of figure 10; time measures confirm the real-time operation of the system and the stability of the reconfiguration logic. Future research directions include the improvement to the middleware design in order to support varying numbers of clients and to better adjust to the requirements of virtualized systems (using the baseline work described in [18] and [12]).

Acknowledgement

This research was partly supported by iLAND (EU ARTEMIS-1-00026) granted by the ARTEMIS JU and the Spanish Ministry of Industry, Commerce and Tourism. It has also been partly funded by REM4VSS (TIN2011-28339) and M2C2 (TIN2014-56158-C4-3-P) project grants of the Spanish Ministry of Economy and Competitiveness.

7. REFERENCES

[1] Apache Software Foundation. *Jini^{TM} network technologies specification. Apache River v2.2.0.* 2013. `https://river.apache.org/doc/spec-index.html`

[2] D. Arney, J. Plourde, R. Schrenker, P. Mattegunta, S. F. Whitehead, and J. M. Goldman. *Design Pillars for Medical Cyber-Physical System Middleware* Medical Cyber Physical Systems – Medical Device Interoperability, Safety, and Security Assurance (MCPS). Dagstuhl Publishing. 2014.

[3] ASTM International. *ASTM F2761 – Medical Devices and Medical Systems – Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE).* 2009.

[4] J. Cano, M. García-Valls. *Scheduling component replacement for timely execution in dynamic systems.* Software: Practice and Experience, vol. 44(8), pp. 889-910. August 2014.

[5] N. Deakin. *JSR 343: Java^{TM} Message Service 2.0.* Oracle. March 2013.

[6] M. García-Valls, P. Uriol-Resuela, F. Ibánez-Vázquez, P. Basanta-Val. *Low complexity reconfiguration for data-intensive service-oriented applications.* Future Generation Computer Systems, vol.37. July 2014.

[7] M. García Valls, P. Basanta-Val. *A real-time perspective of service composition: Key concepts and some contributions.* Journal of Systems Architecture - Embedded Systems Design, vol. 59(10), pp. 1414–1423. November 2013.

[8] M. García Valls, P. Basanta Val. *Usage of DDS data-centric paradigm for remote monitoring and control laboratories.* IEEE Transactions on Industrial Informatics, vol. 9(1). February 2013.

[9] M. García-Valls, F. Ibánez-Vázquez. *Integrating Middleware for Timely Reconfiguration of Distributed Soft Real-Time Systems with Ada DSA.* Proc. of 17^{th} International Conference on Realiable software technologies and applications – Ada-Europe 2012, pp. 35-48. Stockholm, Sweden. 2012.

[10] M. García Valls, T. Cucinotta, C. Lu. *Challenges in real-time virtualization and predictable cloud computing.* Journal of Systems Architecture, vol.60(9), pp736–740. 2014.

[11] M. García Valls, P. Basanta-Val. *Analyzing point–to–point DDS communication over desktop virtualization software.* Computer Standards & Interfaces 49, pp.11-21. January 2017.

[12] M. García Valls, C. Calva-Urrego, A. Alonso, J. A. de la Puente. *Adjusting middleware knobs to assess scalability limits of distributed cyber-physical systems.* Computer Standards & Interfaces. DOI: 10.1016/j.csi.2016.11.003 2017.

[13] M. García Valls, R. Baldoni. *Adaptive middleware design for CPS: Considerations on the OS, resource managers, and the network run-time.* Proc. 14^{th} Workshop on Adaptive and Reflective Middleware (ARM). Co-located to ACM ACM/IFIP/USENIX Middleware. Vancouver, Canada. December 2015.

[14] M. García-Valls, L. Fernández Villar, I. Rodríguez López. *iLAND: An enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems* IEEE Transactions on Industrial Informatics, vol. 9(1), pp. 228-236. February 2013.

[15] M. García-Valls, P. Basanta-Val, I. Estévez-Ayres. *Real-time reconfiguration in multimedia embedded systems.* IEEE Transactions on Consumer Electronics, Vol. 57, No. 3, pp. 1280-1287. August 2011.

[16] M. García-Valls, L. Fernández Villar, I. Rodríguez López. *iLAND project.* Sourceforge. https://sourceforge.net/projects/iland-project/ Last accessed, 2016.

[17] M. García-Valls, A. Alonso, and J.A. de la Puente. *A Dual-Band Priority Assignment Algorithm for QoS Resource Management.* Future Generation Computer Systems, vol. 28(6), pp. 902–912. June 2012.

[18] M. García-Valls. *A proposal for cost-effective server usage in CPS in the presence of dynamic client requests.* In Proc. of 19^{th} International Symposium on Real-Time Computing (ISORC). York, UK. May 2016.

[19] J. Hatcliff, A. King, I. Lee, A. MacDonald, A. Fernando, M. Robkin, E. Vasserman, S. Wininger, and J. M. Goldman. *Rationale and architecture principles for medical appliction platforms.* In Proc. of 3^{rd} IEEE/ACM Conference on Cyber-Physical Systems (ICCPS). 2012.

[20] H. Kopetz, G. Bauer. *The time-triggered architecture.* Proceedings of the IEEE 91 (1), 112-126. 2003.

[21] R. Martins, L. Lopes, F. Silva, P. Narasimhan. *Stheno, a real-time fault-tolerant P2P middleware platform for*

light-train systems. Proc. of 28^{th} ACM Symposium on Applied Computing (SAC). 2013.

[22] ISO/IEC Information Technology Task Force (ITTF). *OASIS AMQP1.0 – Advanced Message Queuing Protocol (AMQP), v1.0 specification*. I(SO/IEC 19464:2014). 2014.

[23] J. Oliveira, J. Pereira. *Experience with a middleware infrastructure for service oriented financial applications*. Proc. of 28^{th} ACM Symposium on Applied Computing (SAC), pp. 479–484. 2013.

[24] Object Management Group. *The Common Object Request Broker. Architecture and Specification, Version 3.3*. http://www.omg.org/spec/CORBA/3.3 (on-line access) November 2012.

[25] Object Management Group. *A Data Distribution Service for Real-time Systems Version 1.4*. http://www.omg.org/spec/DDS/ (on-line access) April 2015.

[26] Object Management Group. *A Data Distribution Service for Real-time Systems Version 1.2*. 2007.

[27] P. Pedreiras, P. Gai, L. Almeida, G. C. Buttazzo. *FTT-Ethernet: a flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems*. IEEE Transactions on IIndustrial Informatics, vol. 1 (3), pp. 162-172. 2005.

[28] I. Rodríguez-López, M. García-Valls. *Architecting a Common Bridge Abstraction over Different Middleware Paradigms*. Proc. of 16^{th} International Conference on Realiable software technologies and application, pp.: 132-146. Edimburgh, UK. 2012.

[29] R. Schantz, et al. *Towards adaptive and reflective middleware for network-centric combat systems*. Encyc. of Software Engineering. Wiley&Sons. 2002.

[30] S. Slichting, S. Polhsen. *An architecture for distributed systems of medical devices in high acuity environments. A Proposal for Standards Adoption*. Drager. 2014.

[31] Sun Microsystems. *Java^{TM} Remote Method Invocation API*. http://docs.oracle.com/javase/7/docs/technotes/guides/rmi/ (on-line access). 2016.

[32] ZeroC Inc. *The Internet Communications Engine*. http://www.zeroc.com/ice.html (on-line access). 2003.

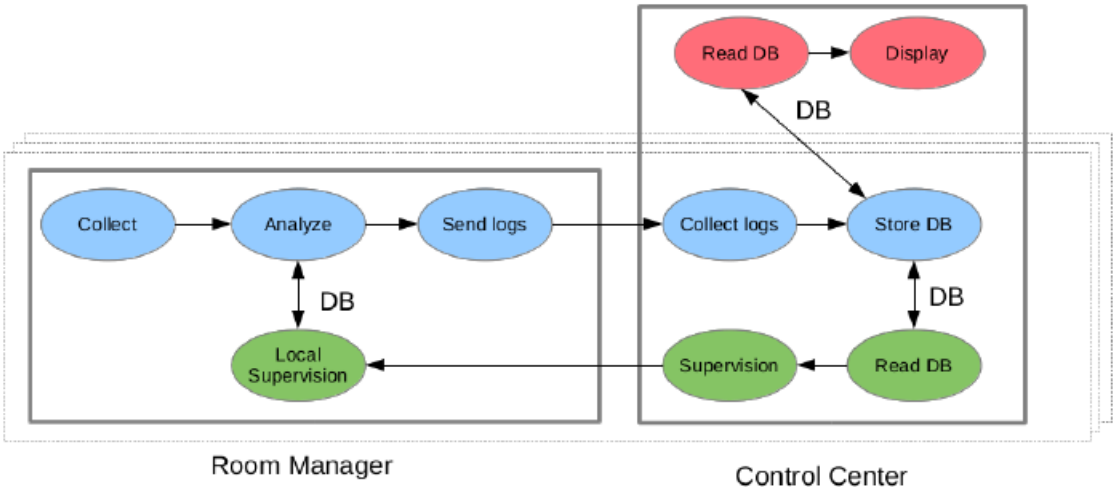

Figure 9: Service deployment in the elderly house monitoring system

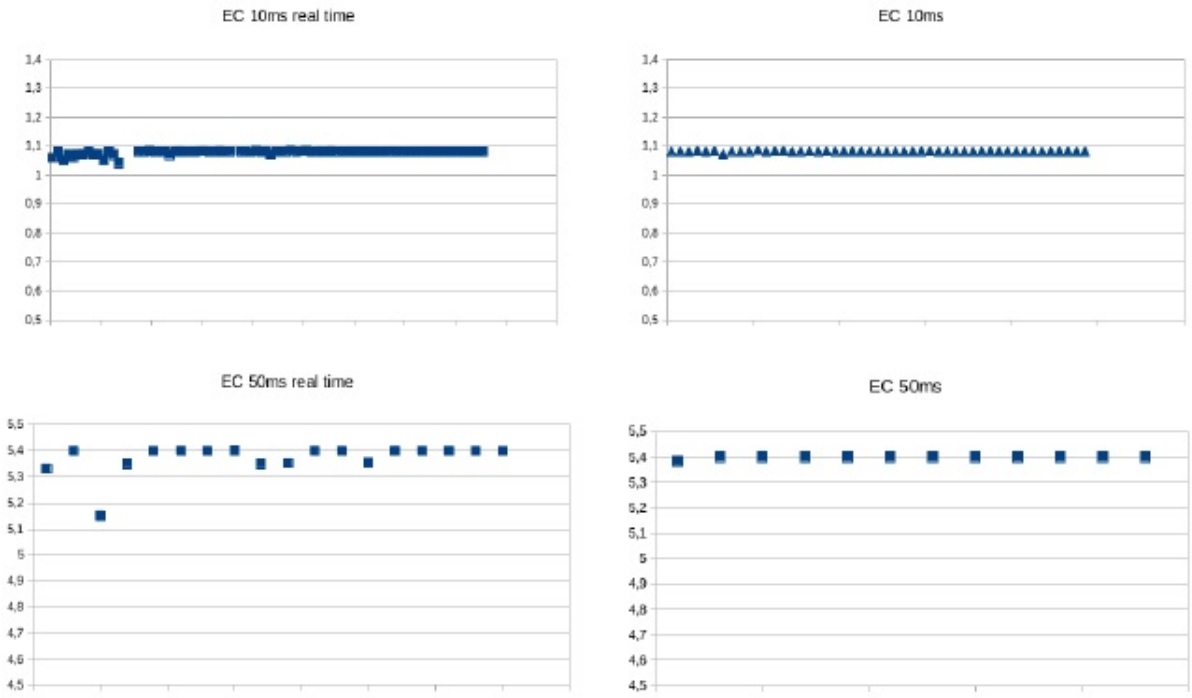

Figure 10: Experimental results over time triggered communication: Reconfiguration times