A scalable, self-adaptive architecture for remote patient monitoring

Andreas Polze
Operating Systems and Middleware
Hasso-Plattner-Institute at University Potsdam
P.O.Box 90 04 60, D-14440 Potsdam
andreas.polze@hpi.uni-potsdam.de

Peter Tröger
Operating Systems and Middleware
Hasso-Plattner-Institute at University Potsdam
P.O.Box 90 04 60, D-14440 Potsdam
peter.troeger@hpi.uni-potsdam.de

***Abstract*— Density of medical clinics, ease of access to doctors, the age pyramid of the population - these are differentiating factors among patients living in urban or rural areas which may affect the probability to survive certain diseases and attacks dramatically. The German Fontane project, a collaboration of medical experts, IT researchers and companies, focuses on remote monitoring and aftercare facilities for stroke patients and patients with heart diseases in rural Brandenburg.**

Within this paper, we discuss central requirements on the object-oriented self-adaptive middleware being developed for Fontane. We describe the overall system requirements, and show our approach for combining real-time data streaming interaction with classical object-oriented remote calls in one distributed middleware stack. The resulting system will form the foundation for transmission of real-time data from devices at patients' home to the remote tele-medicine monitoring center.

I. INTRODUCTION

Project Fontane is a collaborative research effort by a consortium of 20 partners, among them the heart specialists at Charité, the German Telekom as a mobile network provider, various hospitals and health insurance companies, as well as device manufacturers for personal medical devices. The project is intended to increase the quality of medical care for rural areas based on information technology for remote monitoring, medical innovation (such as biomarkers), and medical process innovation (such as the "shop-in-shop cardiologist"). The consortium has teamed up in order to build a pilot remote patient monitoring system and obtain usability information through three different medical studies with more than 5000 patients.

The Operating Systems and Middleware Group at Hasso-Plattner-Institute is part of the Fontane consortium and responsible for the architecture of a rule-based self-adaptive middleware for transmittal of medical data from the patient´s home to a tele-medicine monitoring center. Within this paper, we describe the design rationales and architectural decisions for the Fontane system.

The Fontane project builds upon experiences from the predecessor project called "partnership-for-the-heart" [9] which solved the monitoring problem for 500 patients in the area of Berlin, Germany. However, in contrast to the predecessor project and with more about 3-5000 patients under monitoring, Fontane will face scalability issues that require new approaches for prioritizing the transmission and display of data. Another novelty of our proposed architecture will be its generality: thanks to its self-adaptivity and its rule-based operation it will be applicable not only to monitoring of heart patients but to those suffering from other diseases as well.

II. ARCHITECTURAL CHALLENGES

The Fontane project is driven by a detailed requirement specification from the medical experts in the consortium, which based on daily live experience with existing surveillance technologies. Another influencing factor for the middleware software requirements are technological components provided by the medical equipment manufactures in the consortium.

The Fontane software architecture works in general as scalable connectivity platform between medical devices at a patients home, and a central tele-medicine center located at the hospital site (see Figure 1).

Patients at home are equipped with personal medical devices, such as a high resolution scale, a blood pressure meter, a mobile electrocardiogram, or a biomarker analyzer. Devices may create data sporadically - i.e.; sending one data set as a result of a single measurement triggered by the patient - or periodically as a data stream over a period of time. All home devices communicate via Bluetooth or a similar technology with a central data hub, namely the **home broker**. It implements a store-and-forward scheme for the measured medical data, which are ultimately sent to the tele-medicine center.

The tele-medicine center has to manage data from about 5000 patients living in rural area in northern Brandenburg, Germany. Medical data is stored in an electronic patient's file. Doctors at the tele-medicine center are monitoring patient data on a 24x7 basis. However, due to the sheer number of patients, data have to be ordered according to their criticality. Doctors will diagnose patients and store their diagnostic reports in the system. Based on the diagnosis, further measures such as making an appointment with the patient or sending out emergency team to the patient have to be taken.

The project on which this report is based was funded by the Ministry of Economy of Brandenburg and the EU under grant No. 80136574. This publication reflects the views only of the author.

The tele-medicine center operation relies on a business intelligence application steered by a rule base, in order to infer criticality information from recent diagnostic reports and apply this information to current data sets. The rule-based implementation allows the system to implement self-adaptive and prioritizing behavior with respect to the incoming data.

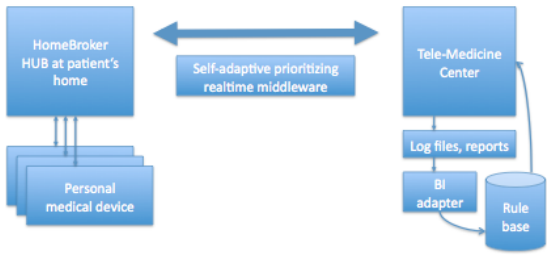

Figure 1: Fontane architecture

The Fontane architecture has to be scalable at three different positions in both hardware and software, namely in the home broker, the transmission middleware, and the tele-medicine center.

The home broker architecture has to be extensible in order to allow for integration of new devices that are currently not on the market. With each device, a receiver task inside the home broker is associated. Some of these devices, i.e.; an electrocardiogram, will deliver real time data streams over a period of time. The home broker internally has to support periodic and sporadic real-time tasks for the data gathering from different connected devices. Connected devices are likely to be changing over time, which demands some dynamic real-time scheduling approach for the different driver modules. At the same time, the home broker has to implement standardized middleware communication protocols over radio links (e.g. EDGE, GPRS or UMTS) in order to allow for interaction with the tele-medicine center. One specialty is the peer-to-peer communication mode – not only the home broker is initializing communication attempts with the telemedicine center, but also the doctors at the center must be able for data pulling or online surveillance activities for a chosen patient.

Scalability at the home broker communication layer concerns the transmission capacity (bandwidth) available inside a radio cell, and the actual low level network protocol negotiated between home broker and network. In order to allow bi-directional communication and remote object invocations initiated from home broker side as well as from the tele-medicine center, one possible approach is a keep-alive mechanism for home broker – initiated connections.

Scalability at the level of the tele-medicine center is mainly concerned with the number of communication partners. Since the home broker implements store-and-forward and since typical transmissions are either remote object invocations (for transmitting scalar data, such as a weight or blood pressure values) or real-time data streams of restricted durations (such as a 2-5 minute measurement of a electrocardiogram), it is planned to realize a dynamic client side call-admission scheme. By partioning the available communication channel dynamically, e.g. with respect to the current link error and data transmission rate, the resulting middleware can offer a self-adaptive optimization of parallel high and low priority data transmissions. We plan to rely upon related research results for data transfers under timing constraints in mobile phone networks [10].

Another aspect is a predictable timing behavior for the real-time server objects in the tele-medicine center. This is especially needed for rendering streamed medical data in a live surveillance operation. The Fontane middleware therefore needs to fulfill both an optimal scalability in the number of supported communication partners, and prioritized processing of real-time streaming data transmissions. Both non-functional aspects shall be realized with standardized middleware technologies, in other to result in a product quality and sustainable software stack.

Dependability aspects of the Fontane middleware stack mainly relate to the clear identification of partial failures in the communication part. The middleware is expected to identify at least timing and crash faults [1] of connected home brokers, in order to give a clear indication of data validity for the monitoring doctor in the tele-medicine center. The store-and-forward approach in the home broker, combined with according meta-data information such as validity time and checksums, enables a straightforward fault recovery in the resulting middleware stack.

III. INITIAL APPROACH

The tele-medicine center implementation is primarily based on a given stack of standardized medical software, such as data analysis and visualization tools and a commercial-off-the-shelf (COTS) implementation of an electronic health record. The Fontane middleware stack is therefore expected to deliver soft real-time communication facilities based on COTS operating systems. Even though such systems do not provide timing guarantees in their operation, the resulting setup has to fulfill soft real-time characteristics for the display of streaming data (from a portable electrocardiogram device at patient's home) in a period of 2-5 minutes. Following a divide and conquer approach, we use a mashup concept to integrate the viewer´s user interface with the electronic health record, but manage the scheduling of the viewer thread(s) completely independently from the existing medical software.

Due to the expected combination of soft real-time demands with standardized communication facilities, we decided for the architectural extension of standard CORBA 2.x middleware. Dependability aspects will be tackled by the implementation of FT-CORBA compliant fault detectors and analyzers.

In order to solve the discussed timing issues with high and low priority communication, we perform a demultiplexing of real-time and non-realtime traffic in the tele-medicine center.. This is realized by two architectural patterns, which are at the heart of our Fontane architecture. The first pattern is 'composite objects', an approach for 'timing firewalls' between non-real-time and real-time parts in a CORBA object implementation. The second pattern is

the ‘scheduling server’, which enables the non-intrusive partitioning of operating system resources for different tasks.

IV. Composite Objects – An Architectural Pattern for the Data Broker at the Patient's Home

The Fontane home broker has to fulfill a number of tasks. Among them is the retrieval of real-time monitoring data streams from medical devices such as a portable electrocardiogram, the retrieval of sporadic non-real-time data from devices such as a scale, the secure local storage of these data, control flow interaction with the tele-medicine center (e.g. device status updates), as well as the real-time transmission of measurements to the tele-medicine center.

The crucial aspect of the home broker implementation is the combination of real-time interfaces for device communication, real-time data transmission with the tele-medicine center, and high-level middleware interfaces for non-critical remote object interaction. This includes the discussed consideration of QoS enabling schemes for the radio connection to the tele-medicine center. Our architecture for the home broker considers this timing behavior duality of the home broker objects by an appropriate design pattern for CORBA objects in the home broker.

Our concept of *Composite Objects* is an approach to integrate responsive (fault-tolerant, real-time) and CORBA computing into a single object-based framework. *Composite Objects* allow the programmer to make an explicit tradeoff between an application’s predictable resource utilization and its communication latency. Therefore, they make implementation details visible and explicit which are usually hidden and abstracted away by CORBA. We advocate a ‘"co-existence and the-need-to-know" approach to reach this goal, by applying the following design rules:

a) NON-INTERFERENCE: General purpose object functionality and time-critical fault-sensitive object functionality will not burden each other.

b) INTER-OPERABILITY: The services exported by general purpose computing objects and by responsive (real-time, fault-tolerant) computing objects can be utilized by each other.

c) ADAPTIVE ABSTRACTION: Lower level information and scheduling actions needed by responsive object parts are available for real-time objects but transparent to non-real-time objects.

Our implementation strategy for ‘*Composite Objects*’ in Fontane follows these design rules. Figure 2 shows the overall structure of such objects. In terms of CORBA object implementations, they provide functionality to react on remote method invocations, so they can be seen as descendants of a class which implements the typical Object Adaptor functionality (Portable Object Adaptor Implementation (POA) in CORBA jargon). On the other hand, *Composite Objects* have the capability to create real-time programming abstractions like prioritized threads and real-time communication channels. Thus, they can be seen as descendants of a class, which implements real-time servers.

Composite Objects consist of a real-time and a non-real-time part. Design time and runtime guarantees can be given for execution of real-time methods. In contrast, methods in the non-real-time part of the composite object are executed following a best effort approach. Those non-real-time methods correspond to the well-known standard CORBA method invocations. It is the task of the *Composite Object* to ensure that the service to non-real-time clients will be carried out in best effort without adversely affecting the real-time computation.

The principle of ADAPTIVE ABSTRACTION as realized here allows hiding all implementation details from the CORBA user, whereas scheduling and timing information is accessible for the real-time part of such an object.

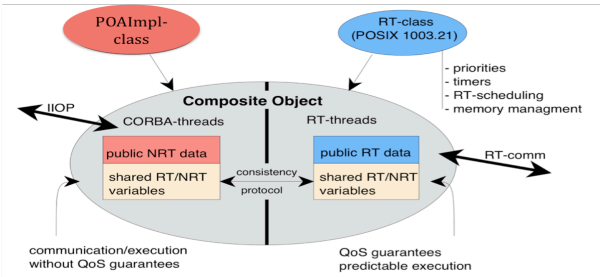

Figure 2: Composite Object structure

Composite Objects establish timing firewalls [3] between real-time and non-real-time (CORBA) computing, so that the non-real-time part cannot violate the real-time scheduling rules [6] that are needed by the real-time part. This approach allows us to implement the idea of NON-INTERFERENCE. Additionally, *Composite Objects* allow for INTER-OPERABILITY between real-time and non-real-time computing by providing data transfer mechanisms in such a way that real-time accesses to the data cannot be blocked by non-real-time functions.

Data replication is the key to independent data accesses from real time and non-real time threads inside a *Composite Object*. We split an object’s data into two parts. One part is statically locked in memory to fulfill real time scheduling assumptions (RT data), the other part is treated like a standard non-real time user’s process’ data (NRTdata). A pair of RT/NRTvariables can be viewed as replicated variable. *Composite Objects* implement a consistency protocol to update both replicas and create the impression of a shared variable. Those data structures are accessible by both, real time and non-real time threads and allow for predictable interactions. Therefore, a *Composite Object* is rather a design concept than a contiguous data structure at runtime of a program.

Our implementation of the Fontane home broker will employ these special objects in order to separate real-time threads for medical devices, real-time threads for streaming of medical data, and non-real-time worker threads from each

other, even though all of them operate on the same set of measurements. A role-based access control model controls access to both types of interfaces. Besides the patient and the doctor at the tele-medicine center, our role model has to implement special roles for access to data stored in the home broker for emergency teams (full access) and nurses (restricted local access). This can be understood as special case of remote data access, where local data reading facilities have to be prioritized over the remote surveillance functionality.

V. SCHEDULING SERVER - PREDICTABLE RESOURCE MANAGEMENT AT THE TELEMEDICINE CENTER

The decoupling of real-time and non-realtime processing on standard operating systems is one of the challenges for the tele-medicine center part in Fontane. Patient´s data today is stored inside a so-called electronic patient´s file. There exist a number of commercial off-the-shelf implementations of those files. In order for our implementation to integrate with the electronic patient´s file, our system has to support a given set of libraries and data formats for medical data encoding (e.g. HL7, CDA, EN 13606). In addition, a plugin-based architecture is being used to display medical data and diagrams in standard web browsers to doctors. In order to fit with the typical working environment of these non-technical end users, the Fontane solution has to be applicable to standard IT systems. More specifically, both technical issues demand the implementation of soft real-time resource partitioning in a commercial-off-the-shelf operating system.

The Fontane implementation at the tele-medicine center employs a rule-based system in order to prioritize and order monitoring data received from different patients. Beside long-term response activities of doctors to a single measurement (order of hours), the system also has to display real-time data streams from electrocardiogram measurements that have timing requirements in the order of milliseconds. While a best-effort approach is taken for the first class of requirements, real-time scheduling algorithms and deterministic resource reservation are applied on the level of the operating system for the time-critical demands. Another challenge concerns the existence of third-party code which cannot directly be controlled from Fontane but which must be utilized as part of the center software stack. In order to implement predictable resource management on a standard operating system, we will apply our previously developed *Scheduling Server* [5] as underlying architectural pattern.

Today's off-the-shelf operating systems implement a number of different scheduling policies, among them fixed-priority scheduling. Threads executed under that policy do not experience aging and keep control over the processor until their adjustable quantum (time slice) expires and a higher-priority thread becomes ready. The priority range for the fixed scheme is typically above the default scheduling priorities of user and non-IRQ related kernel mode threads. Furthermore, systems like Linux, Windows or Mach implement systems calls, which allow manipulation of a thread´s priority.

The combination of both features allows for the implementation of the *Scheduling Server* dispatcher thread, which runs at the highest priority available for user mode threads on the system. The server usually sleeps, thus allowing normal operation of the system. Time-critical threads in the Fontane tele-medicine center software may register as clients with the scheduling server. The scheduling server using a middleware-defined priorization scheme then schedules these threads. Registered threads are either running at a low priority or are even suspended, leaving the rest of the system and a interactive workload undisturbed. Based on configured scheduling setting, one of the client threads is picked and raised to the second-highest priority in the system. The *dispatcher thread* then suspends itself, giving all the CPU cycles to the selected Fontane server thread. When the suspend phase is over, the operating system again activates the dispatcher thread, which resets the Fontane server's thread priority and gives the rest of the system a chance to run again.

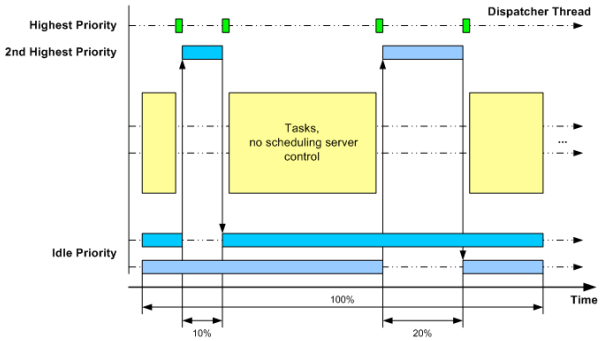

Figure 3: scheduling server approach

Figure 3 demonstrates how priorities are manipulated by the dispatcher thread. One may notice, that kernel and interactive threads (those not belonging to Fontane) are executed within fixed intervals - independent of the number of parallel tasks. The *Scheduling Server* in it's current implementation (for POSIX and Windows systems) currently treats all Fontane server threads as having the same priority. An slight extension of the existing realization will allow the assignment of different priorities under a chosen static real-time scheduling approach (e.g. EDF, RMS). This enables the Fontane middleware to dynamically prioritize different home broker data processing tasks, based on the dynamic configuration from the end-user influenced rule set.

VI. RELATED WORK

The combination of object-oriented CORBA middleware with both real-time and fault tolerance capabilities is being investigated mostly in the context of defense and aircraft management systems [2] and has resulted in the real-time CORBA specification [4]. Pioneering work in the field of object-oriented real-time computing has been carried out by Kane Kim with his TMO (Time-triggered Message-triggered Objects) approach [7][8].

With our work in the Fontane project, we plan to build upon these existing research efforts. However, additional

research is needed in order to extend object-oriented real-time computing with support for predictable communication via (unreliable) radio links (i.e.; Edge, GPRS, UMTS).

Different operating systems support the predictable CPU resource partitioning as part of their scheduling functionality. Beside the traditional real-time scheduling policies in POSIX-compliant Unix systems, such as Solaris and HP-UX, Microsoft Windows (Vista and beyond) now also implement a Multi-Media Class Scheduler (MMCS) for soft-realtime multimedia transmissions. Mac OS X implements the concept of CPU reserves as another approach to predictably share CPU resources among real-time (e.g. iTunes) and non-realtime threads. We plan to investigate these recent developments in comparison with the Scheduling Server approach, especially with respect to scheduling overhead and latency.

VII. Conclusion and Outlook onto further research

Within this paper, we have discussed the initial setup, requirement specification, and an architectural sketch for the forthcoming implementation of the Fontane system for remote patient monitoring. We have presented the overall architecture consisting of a home broker (hardware + software) at the patient's home, a self-adaptive prioritizing middleware layer, and a rule-based server engine at the tele-medicine center.

Genericity, scalability, and extensibility are key design requirements for all system components. The application of off-the-shelf hardware and software was a pre-requisite for our Fontane system architecture. Being based on a standard foundation, we rely on the "Composite Objects" and "Scheduling Server" architectural patterns as well as on Real-Time CORBA middleware in order to fulfill timing and reliability requirements for the Fontane system. We are currently implementing a prototype system in a testbed at Hasso-Plattner-Institute at University Potsdam.

The first field study with patients in northern Brandenburg will start July 1, 2010.

VIII. References

[1] Cristian, Flavin: Understanding Fault-Tolerant Distributed Systems. In: Communications of the ACM 34 (1991), Nr. 2, S. 56-78

[2] Gokhale, Aniruddha S.; Natarajan, Balachandran; Schmidt, Douglas C.; Cross, Joseph K.: Towards Real-Time Fault-Tolerant CORBA Middleware. In: Cluster Computing 7 (2004), Nr. 4, S. 331-346

[3] Polze, Andreas; Sha, Lui: Composite Objects: Real-Time Programming with CORBA. In: EUROMICRO '98: Proceedings of the 24th Conference on EUROMICRO. Washington, DC, USA : IEEE Computer Society, 1998

[4] Schmidt, Douglas C.; Kuhns, Fred: An Overview of the Real-Time CORBA Specification. In: Computer 33 (2000), Nr. 6, S. 56-63

[5] Schöbel, Michael; Polze, Andreas: Kernel-mode scheduling server for CPU partitioning: a case study using the Windows research kernel. In: SAC '08: Proceedings of the 2008 ACM symposium on Applied computing. New York, NY, USA : ACM, 2008, S. 1700-1704

[6] Sha, Lui; Rajkumar, R.; Gagliardi, M.: Evolving Dependable Real-Time Systems. In: 1996 IEEE Aerospace Applications Conference. Proceedings. Aspen, CO : IEEE New York, NY, USA, 1996, S. 335-46

[7] Kim, K.H., "Object Structures for Real-Time Systems and Simulators", IEEE *Computer*, August 1997, pp.62-70.

[8] Kim, K.H., "APIs for Real-Time Distributed Object Programming", IEEE Computer, June 2000, pp.72-80.

[9] Köhler, F.; Schierbaum C.; Konertz W.; Winter S. „Partnership for the Heart“. German-Estonian Health Project for the Treatment of Congenital Heart Defects in Estonia. Health Policy 2005; 73(2): 151-159

[10] Chakravorty R.; Banerjee S.; Rodriguez P.; Chesterfield J.; Pratt I.: Performance Optimizations for Wireless Wide-Area Networks: Comparative Study and Experimental Evaluation, Proceedings of ACM Mobicom 2004.