# Reducing communication-related complexity in heterogeneous Networked Medical Systems considering non-functional requirements

Morteza Hashemi Farzaneh, Suraj Nair, Mohammad Ali Nasseri, Alois Knoll
Institute of Robotics and Embedded Systems
Technische Universität München
85748 Garching bei München, Germany
**Email: {hashemif, nair, ali.nasseri, knoll}@in.tum.de**

***Abstract*—Networked Medical Systems (NMS) promise better data exchange in medical infrastructures such as operating rooms in hospitals and clinics. However, the heterogeneous interfaces of medical systems and varied requirements on NMS such as real-time constraints, increase the communication complexity considering network architectures, communication protocols and software/hardware components. In this paper, a robot-assisted eye surgery is used as a clinical use case. Based on this use case and its communication types, non-functional requirements on NMS are derived.**
**An approach for abstraction is proposed which targets at reducing the communication-related complexity in NMS. Complexity reduction in this case means that a multi-interface middleware in NMS abstracts the detailed knowledge required for implementation of different communication types such as real-time communication. The middleware architecture is divided into two main parts: Communication Abstraction Provider (CAP) and Communication Abstraction Bridge (CAB). CAP is the central component of the middleware which connects the medical systems using CABs.**
**In this paper, the focus is on the complexity reduction of the real-time communication part. For this purpose, real-time communication protocols are investigated and evaluated for application in the CAP/CAB architecture. The result of the evaluation shows that Ethernet POWERLINK is the most suitable real-time communication protocol for the CAP/CAB architecture.**

## I. Introduction

Networked Medical Systems (NMS) offer higher flexibility in exchanging data between medical systems. A NMS is defined as a network with the following node types: 1- Medical systems, which provide one or more interfaces for communication with the rest of the network, 2- IT-systems which are used by medical systems to store and restore medical data of medical systems [1], 3- Middle-ware, which is used by medical systems and IT-systems to communicate with each other [2]. Depending on the medical applications and use cases, various types of data are exchanged in a NMS. For instance, a surgery robot needs motion controlling data from a controller device using a real-time communication. Transmitting high quality video data from a medical device such as an endoscope to a monitor in real-time, is another use case in the operating rooms. Not each communication must real-time. For example, exchanging configuration parameters between medical systems or storing patients' data is done using a non-real-time communication providing high bandwidth. Considering these different use cases, various communication systems can be used. The term *communication system* in this paper refers to a system which includes communication hardware, software and protocol. Examples for common used communication systems for the mentioned use cases are Controller Area Network (CAN) [3] for rapidly and priority based transmitting data, High Definition Multimedia Interface (HDMI) [4] for sending and receiving high quality video data, Transmission Control Protocol/Internet Protocol (TCP/IP) [5] for exchanging non-real-time data between medical systems and IT systems such as Digital Imaging and Communications in Medicine (DICOM) [1] and Picture Archiving and Communication System (PACS) [6].
The heterogeneous interfaces of medical systems and varied requirements on NMS such as real-time constraints, increase the communication complexity considering network architectures, communication protocols and software/hardware components. The high communication complexity in NMS limits the integration of new medical systems into existing NMS. Connecting a new medical system to another medical system, requires deep knowledge about the target medical system which is not always available. Only knowledge about the target medical system is not enough and for implementation of a specific communication e.g. real-time communication, detailed knowledge and experiences in the specific communication protocol, hardware and operating system is required.
In this paper, an abstraction-based approach is proposed for reduction of communication complexity in NMS. A real world use case is used to derive non-functional requirements on NMS which is covered by the proposed approach. This approach includes a middleware consisting of two parts: Communication Abstraction Provider (CAP) and Communication Abstraction Bridge (CAB). CAP is the central component of the middleware which connects the medical systems using CABs. Using CAP and CAB, a developer just has to write the application code of a specific connection between two medical system. This application contains the communication logic and is based on the description of medical systems that have to be connected. The developer does not have to deal with the complex low level implementation details. If an already connected medical system has to be used for another connection, then a developer has to change only the application code without any modification in the lower layers. In this paper, the focus is on the real-time-related part of CAP/CAB approach. The

starting point of implementation of the real-time-related part is investigating and evaluation of available real-time protocols. The paper is structured as follows: In the next section, an eye surgery use case is presented which is the basis for deriving non-functional requirements on NMS. In section III the core components of the CAP/CAB abstraction are explained. The real-time communication protocols are investigated and evaluated in section IV and the implementation challenges are discussed in section V Conclusion and future work are included in the final section VI.

## II. An use case and non-functional requirements on NMS

To derive requirements on NMS, real world use cases are required. In this paper, a robot-assisted eye surgery is used as use case. From the used medical systems in this use case and the type of required communication type between them, the general non-functional requirements on NMS are derived. These requirements are verified comparing other real world use cases[1] with eye surgery use case.

Motion controlling of the surgery robot has to fulfill hard real-time requirements. Non-deterministic delays in controlling robot's actions will have disastrous consequences for the eye under operation. A maximum robot's response time of less than one millisecond has to be guaranteed. Calculating the maximum response time begins with moving the input device (space mouse), generating motion controlling data such as robot position data, sending the generated data to the robot, processing the received data and calculating the new position of the robot and ends with positioning of the robot. All these actions have to be done within 1 millisecond. Because involving a human in this interaction, a maximum response time of 30-60 ms is satisfying (based on the surgeons experiences in the OR.NET project which deals with deals with dynamic networking in operating rooms). But why 1 ms is required? The answer is that for each operating robot, a safety mechanism has to be implemented. For instance, If the surgeon enters dangerous area while operating the eye, a sensor has to recognize that and send stop message to the robot. In this safety mechanism, 1 ms maximum response time of robot after receiving the stop message has to be guaranteed. Else, the surgeon can damage the eye under operation. This value can be higher and more flexible depending on specific use cases but the worst case situation is assumed.

For the eye surgery robot a video feedback systems is planned. The video from the camera is sent to this feedback system. It analyses the incoming video data and give a feedback to the robot. Analyzing this video data helps e.g. the co-surgeon to be sure that an action is done successfully or not. A maximum delay streaming the video must not be exceeded. This value is variable and is also based on surgeons' experiences in the OR.NET project.

Storing or loading of patient data in standard IT infrastructures such as DICOM or PACS, has to be supported. Transmitting these big data to or from these IT systems requires a communication with high bandwidth. These IT-systems are based on Ethernet and TCP/IP technologies. A minimum bandwidth of 1 Gigabit/s or optional 10 Gigabit/s has to be supported.

Wireless communication offers a high flexibility in the network topology [7]. In the eye surgery use case, for each medical system an interface for wireless communication in future is required.

Hot-plugging availability is required if e.g. the robot controller devices is disconnected and reconnected to the NMS. Failure by reconnecting the medical systems means low safety for the operating room while operating. Table I depicts the list of derived non-functional requirements.

TABLE I. Derived non-functional requirements

| | Non-functional requirement |
|---|---|
| 1 | Cycle time $< 1ms$ for motion controlling |
| 2 | Real-time video streaming with an acceptable maximum delay for surgeons |
| 3 | 1 Gigabit/s bandwidth for big data |
| 4 | Wireless communication |
| 5 | Hot-plugging availability |

## III. Reducing communication-related complexity in NMS

In this section, an approach for abstraction is introduced which targets at reducing the communication-related complexity in NMS. The communication-related complexity is defined as the knowledge basis that is required for implementation of a specific communication between two medical systems. For example, in the robot-assisted eye surgery use case, for implementation of the real-time communication between the space mouse and the surgery robot, low level knowledge about real-time operating systems, hardware components (embedded system) and real-time communication protocols is required. Complexity reduction in this case means that a multi-interface middleware in NMS abstracts the detailed knowledge required for implementation of the real-time communication. A medical system manufacturer produces mouse spaces or similar controller devices for robot controlling. This devices support only USB interfaces. Another robot manufacturer produces eye surgery robots with an UART interface. The following concrete steps should be done to use the space mouse with the robot: Firstly, a USB to UART converter is required which supports real-time concerting. Secondly, the USB data should be restructured so that the robot can read it. Thirdly, the protocol converter has to know the meaning of USB data (specific only for this mouse) and has to know what the robots can understand (also only specific for this ). For instance, the mouse produces parameters $X$ and $Y$ with their values. But the robot can only understand the parameters *left* and *write*. For implementation of this scenario deeper knowledge about converter hardware (interfaces, etc.), eventually real-time operating system installed on the converter hardware (for supporting real-time data converting) and the real-time communication protocol (In case that they are not connected directly but through an existing network infrastructure) are required.

If the mouse is required to connect to another robot for controlling it, all mentioned steps should be repeated. Considering NMS and heterogeneous medical systems, this challenge get even bigger. It is imaginable that e.g. the robot has multiple real-time interfaces: one interface for controlling and another interface for connection to a sensor which senses the eye environment and sends stop signal to the robot if the surgeon is in a dangerous area. A interface-to-interface implementation of the real-time connection in NMS (with a lot of involved medical systems in the network) as described above, leads to

[1] See www.ornet.org for more information.

a complex and chaotic network regarding the used hardware for real-time communication, real-time communication protocol and network topology.
A middleware-based approach is proposed for reduction of the mentioned communication complexity. This middleware is divided into two main parts:

- Communication Abstraction Provider (CAP)
- Communication Abstraction Bridge (CAB)

Medical systems use CABs for using the offered interfaces of CAP. CAP is the central component of the middleware which connects the medical systems using CABs. According to the derived non-functional requirements in the previous section, CAP and CAB offer four type of interfaces. CAP and CAB support a real-time communication interface (RT), a wireless communication interface (WLAN), an interface for non-real-time communication interface dealing with big data (NRT) and an interface for real-time video streaming (RT-Video). Similar to real-time communication, the abstraction reduces the complexity of WLAN, Video and NRT communication. This approach will fulfill the derived requirements in the previous section. Figure 1 demonstrates an overview of the introduced approach using the eye surgery use case. The space mouse and the surgery robot are connected to the RT interface of CAP using CAB. A developer has only to define the meaning of the input data for the robot using CAB. Defining the meaning of data means that the developer only has to develop the application logic of e.g. the real-time communication between mouse and the robot. The video camera and the display are connected to the RT-Video interface for transmitting video data. All data available in CAP inclusive the big data can be transmitted to the Internet or other IP-based (Internet Protocol) information systems such as DICOM Server using NRT interface.

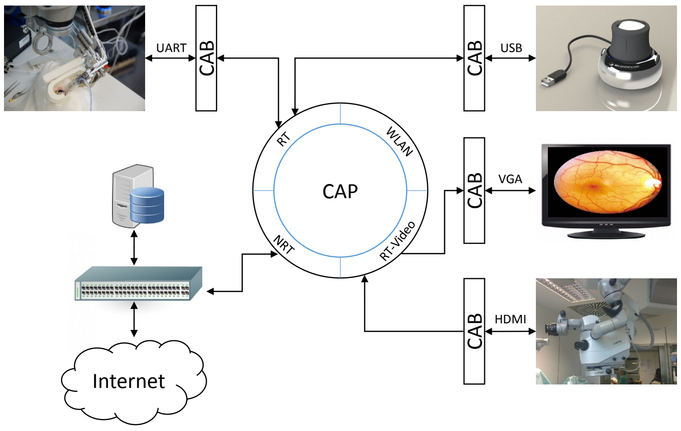


Figure 1. Reducing communication complexity in the eye surgery NMS using Communication Abstraction Provider (CAP) and Communication Abstraction Bridge (CAB)

The RT Unit in CAP deals with organization of real-time communication between different CABs. RT Unit consists of a hardware which is capable for realization of real-time communication. Such a hardware has to support a processor with enough capacity for running the real-time communication protocol. It also has to support multiple interfaces for real-time communication between different CABs. It plays the role of a master which synchronizes the real-time network. On the top of the CAPs' hardware, a real-time operating system is required which has to guarantee the execution of the real-time communication stack using e.g. scheduling priorities. The real-time communication stack runs on the real-time operating system required for deterministic communication between medical systems.
Each real-time medical system has to be connected to a CAB. Each CAB supports a set of heterogeneous interfaces such as USB, UART, etc. Protocol Converter Unit (PCU) is a process in CAB and responsible for mapping the communication interfaces of the top and bottom of CAB. The focus of this paper is on the real-time-related part of the PCU. PCU consists of Real-time Communication Programming Library (RCPL). RCPL is a library which supports a Real-time Application Programming Interface (RTAPI). RTAPI is the interface to the low-level communication with the real-time operating system on CAB. Using RTAPI, access to the Ethernet controller is possible. RCPL offers an abstraction of low-level implementation complexity. In this way, a medical system can open and handle a real-time connection to other medical systems through CAB and CAP for sending and receiving without the need of deeper knowledge about low-level implementation details.
Considering the eye surgery use case, the space mouse is described as a controller device which provides e.g. values for X and Y axis. These values are sent to the robot applying RCPL and RTAPI for a real-time communication. Similar to CAP, each CAB provides at least one real-time interface. Using this interface, data can be received or sent from one medical system to another medical system. CAP plays the role of a master which guarantees the real-time communication between CAB's. Figure 2 offers a look inside the CAP/CAB architecture considering the real-time communication. In this sample and according to the eye surgery use case, medical system 1 (space mouse) is communicating with medical system 2 (surgery robot) in real-time using the CAP and two CABs and the medical systems do not support the same interface (IF-2 and IF-3 refer to USB and URAT). The CAP/CAB concepts does not deal with the inner communication of the medical systems. This concept assumes that the inner communication of a medical system is deterministic so that the communication between a medical system and its CAB is deterministic. This assumption is justified because the medical system provider is aware that its system or device is going to be used for a deterministic communication. If this is not the case, nothing else in NMS can guarantee a deterministic behavior of the medical system. The implementation real-time part of the CAP/CAB architecture consists of three main steps: First, real-time communication protocols have to be investigated and evaluated to select the most suitable protocol. Second, hardware in form of embedded systems have to be investigated and evaluated for implementation of CAP and CAB. The inner components of CAP and CAB (specially PCU and RCPL) have to be implemented. Figure 3 demonstrates the final goal of implementation of CAP/CAB architecture regarding to the real-time communication. Manufactures produce medical systems and their description. The CAP/CAB developing tool offers Hardware, Real-time operation system, a real-time communication protocol stack, PCU on the top of the real-time protocol stack which implements RCPL and RTAPI and an environment for implementation of the communication logic between medical systems. A developer only writes the

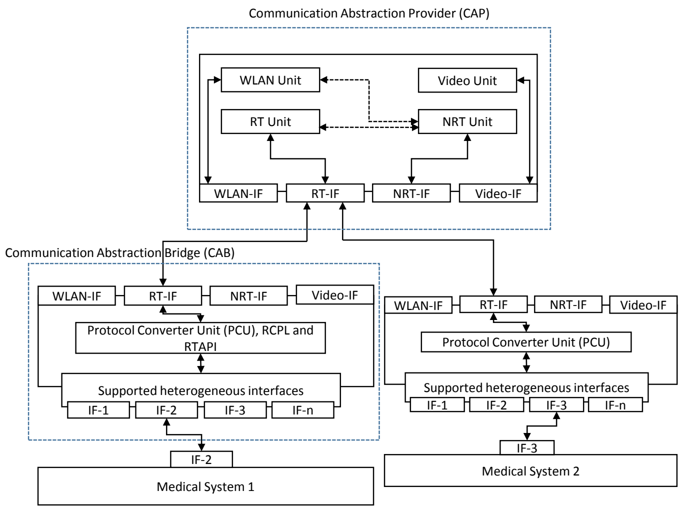

Communication Abstraction Provider (CAP)
WLAN Unit
Video Unit
RT Unit
NRT Unit
WLAN-IF
RT-IF
NRT-IF
Video-IF
Communication Abstraction Bridge (CAB)
Protocol Converter Unit (PCU), RCPL and RTAPI
Protocol Converter Unit (PCU)
Supported heterogeneous interfaces
IF-1
IF-2
IF-3
IF-n
Medical System 1
Medical System 2

Figure 2. CAP and CAB with more details about physical interfaces and the interaction of sub units

application code (the communication logic between medical systems). In the mentioned mouse and robot example, if the same space mouse should be used for real-time communication to another medical system, only the application has to be modified by the developer.

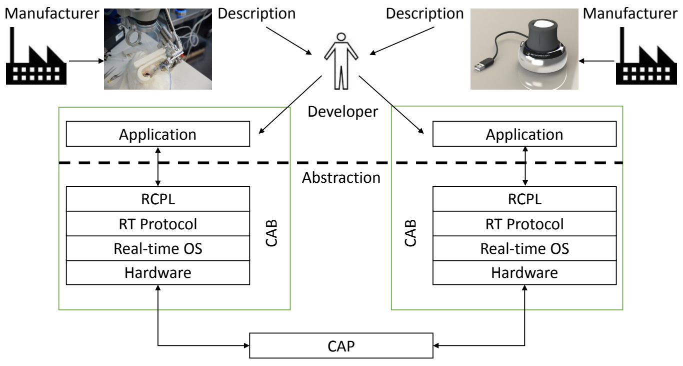


Figure 3. Final goal of CAP/CAB implementation regarding to the real-time communication. Developer uses all available components and writes the application code for two real-time communicating medical systems. The result can be e.g. two boot-able memory cards which can be put into the CABs of the two medical systems. After booting the CABs, the desired real-time communication between two medical systems is available.

## IV. Investigating and Evaluation of Real-Time Communication Protocols

As the first step of implementation of the real-time-related part of the CAP/CAB architecture, real-time communication protocols are investigated and evaluated. Based on the following criteria, the protocols are evaluated:

**Maximum transmission latency:** a maximum transmission latency of less than 1 milliseconds (ms) has to be supported. It means that, if a medical system sends a real-time message to another one then the receiver has to receive this message and handle it within 1 ms. This is defined as the *cycle time* in this paper. This constraints come from the timing requirements of the eye surgery use case. The minimum achievable cycle time of the protocols is evaluated.

**Open source:** in order to support an open source implementation of CAP/CAB, the used real-time protocol has to be open source.

**Hot-plug-ability:** if a real-time connection line is broken (whatever happened) the real-time medical system has to be able to be connected again with a minimum of delay.

**Cost:** the real-time protocol and its components (hardware/software) has to be economical.

**Support:** support is an important factor for implementation and maintenance of the CAP/CAB.

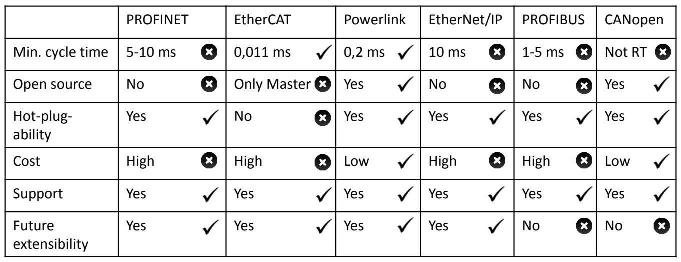
**Future extensibility:** the real-time protocol and its components (hardware/software) has to be extensible in future. For instance, the protocol has to be based on widespread technologies which will exists also in future.

A first investigation of real-time communication protocols shows that the Ethernet-based protocols have advantages over the others. The main reason are low costs and future extensibility. Ethernet is widespread and this leads to lower costs of Ethernet components in comparison to other less widespread protocols. Because of the existence of numerous real-time protocols, only newest (state of the technology) protocols are investigated (4 Ethernet-based and 2 older and not Ethernet-based protocols).

PROFINET [8] is an Ethernet-based real-time communication protocol which support only a minimum cycle time of 5-10 ms. It is not open source and is proprietary protocol provided by Siemens. It provides also hot-plug-ability. Siemens offers a good support for this protocol. Because all components of this protocol are proprietary, its cost is high. It is based on the Ethernet technology and provides high future extensibility.

EtherCAT [9] is an Ethernet-based protocol and supports low cycle time (0,011 ms). It is a proprietary protocol provided by Beckhoff and is not hot-plug-able. There is an open source implementation of EtherCAT Master. Similar to PROFINET the costs are high (specially for Slaves). It also provide high future extensibility because of the Ethernet technology.

POWERLINK [10] is also Ethernet-based and supports a minimum cycle time about 0,2 ms. It is an open source



| | PROFINET | EtherCAT | Powerlink | EtherNet/IP | PROFIBUS | CANopen |
|---|---|---|---|---|---|---|
| Min. cycle time | 5-10 ms ⊗ | 0,011 ms ✓ | 0,2 ms ✓ | 10 ms ⊗ | 1-5 ms ⊗ | Not RT ⊗ |
| Open source | No ⊗ | Only Master ⊗ | Yes ✓ | No ⊗ | No ⊗ | Yes ✓ |
| Hot-plug-ability | Yes ✓ | No ⊗ | Yes ✓ | Yes ✓ | Yes ✓ | Yes ✓ |
| Cost | High ⊗ | High ⊗ | Low ✓ | High ⊗ | High ⊗ | Low ✓ |
| Support | Yes ✓ | Yes ✓ | Yes ✓ | Yes ✓ | Yes ✓ | Yes ✓ |
| Future extensibility | Yes ✓ | Yes ✓ | Yes ✓ | Yes ✓ | No ⊗ | No ⊗ |

Figure 4. Investigation and Evaluation of real-time communication protocols. Powerlink is the winner technology in this evaluation [13], [12], [14], [15], [16], [17].

protocol and hot-plug-able. There is support website with forums and discussions about porting this technology. It is open source and based on the standard Ethernet hardware and its costs are low and it is high extensible in future.

EtherNet/IP [11] (IP means Industrial Protocol) provides a minimum cycle time of 10 ms and is not available as open source. Hence, the costs of the protocol stacks are high. It is hot-plug-able and is because of Ethernet extensible in future. There is no free support of the protocol.

PROFIBUS [12] is the ancestor technology of PROFINET with the difference that PROFIBUS is not based on the Ethernet technology and provides lower bandwidth of 12 Mbit/s but offers a shorter minimum cycle time of 1-5 ms.

CANopen [13] is an communication protocol provided originally by Bosch. The specification of CANopen is free and there is an open source implementation of it. It is a non-cyclic protocol and does not fulfill hard real-time requirements. It supports hot-plug-ability and low hardware costs. Because of the 1 Mbit/s bandwidth, CANopen is not extensible for future. The evaluation result is presented in Figure 4.
The Ethernet-based technology POWERLINK fulfills all requirements and is selected for the further implementation of the rel-time realted part of the CAP/CAB concept.

## V. Implementation challenges

The first step of the implementation of the CAP/CAB real-time-related part with POWERLINK is selecting hardware and real-time operating system for CAB and CAP. Beaglebone Black [18] is selected as the test hardware. The reason for selecting this embedded board are: high computational capacity (1 GHz CPU and 512 MB memory) and numerous communication interfaces such as UART, Ethernet, USB, HMDI, etc.
QNX [19] and Xenomai [20] are selected as test real-time operating systems. QNX is in comparison to Xenomai a microkernel real-time operating system. The main challenge implementing the test real-time communication on Xenomai is the required real-time driver for the Ethernet controller. POWERLINK only supports real-time drivers for a limited number of Ethernet controllers out of the box. The Ethernet controller of Beaglebone Black is not supported.
For soft real-time communication the standard Ethernet driver is enough using the POWERLINK stack in the user space. In this case, increasing the protocol stack priority leads to low latency. But for a hard real-time communication, running the protocol stack in the user space is not satisfying. Because a minimum cycle time of 1 ms is not supported in the user space. To overcome the challenge of writing real-time driver for the Ethernet controller of Beaglebone Black, another test scenario is in progress.
In the microkernel QNX architecture, although all processes and drivers are in the user space, hard-real time processing is guaranteed because of its microkernel architecture.

## VI. Conclusion and future work

Networked Medical Systems (NMS) offer more flexibility exchanging data in medical infrastructures. Heterogeneous interfaces in NMS and non-functional requirements on it, increase the communication complexity from the application layer down to the lowest hardware layer. An eye surgery use cases is used to derive the non-functional requirements on NMS.
An abstraction-based approach for reducing the communication complexity in NMS is proposed. It abstracts the application layer from the lower layers. In this paper, the focus is on the real-time-related part of the approach. For this purpose, a set of existing real-time communication protocols have been investigated and evaluated. The result demonstrated that POWERLINK is the most suitable protocol for application in the proposed approach. The implementation challenges considering POWERLINK on the test hardware Beaglebone Black are discussed.
Future work will focus on further investigation and evaluation of the components of the proposed approach such as suitable hardware for CAP/CAB and real-time operating systems. Simultaneously, PCU and RCPL will be developed based on the lower layers. The proof of the concept will be a demonstrator which includes: a NMS supporting a simplified real-time communication using the proposed abstraction-based approach in this paper. The final step will be developing of other three communication types: WLAN, NRT and Video streaming.

## Acknowledgment

This paper is funded by the German Ministry for Education and Research (BMBF).

## References

[1] S. Horii, "An Introduction to the ACR-NEMA Standards," *The Second International Conference on Image Management and Communication (IMAC) in Patient Care: New Technologies for Better Patient Care*, pp. 235–249, 1991.

[2] F. Kart, G. Miao, L. E. Moser, and P. M. Melliar-Smith, "A Distributed e-Healthcare System Based on the Service Oriented Architecture," in *IEEE International Conference on Services Computing (SCC 2007)*, no. Scc. Ieee, 2007, pp. 652–659.

[3] Q. Du, "Study on distributed control system for a medical robot," in *Automation and Logistics, 2008. ICAL 2008.* Ieee, 2008, pp. 1678–1682.

[4] A. Goertzen, J. Thiessen, X. Zhang, C.-Y. Liu, E. Berg, D. Bishop, P. Kozlowski, F. Retiere, V. Sossi, G. Stortz, and C. Thompson, "Application of hdmi x00ae; cables as an mri compatible single cable solution for readout and power supply of sipm based pet detectors," in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE*, 2012, pp. 3184–3188.

[5] H. Andoh, K. Watanabe, T. Nakamura, and I. Takasu, "Network health monitoring system in the sleep," in *SICE 2004 Annual Conference*, vol. 2, 2004, pp. 1421–1424 vol. 2.

[6] F. Breu, S. Guggenbichler, and J. Wollmann, "PACS: Personal Access Communications System - A Tutorial," *Vasa*, no. June, pp. 32–43, 2008.

[7] S. Lee, U. Kim, K. Kwon, W. Seo, and J. Choi, "Design of on-body antenna for wireless body area network," in *Advanced Communication Technology (ICACT)*, 2012, pp. 300–303.

[8] P. Ferrari, A. Flammini, F. Venturini, and A. Augelli, "Large profinet io rt networks for factory automation: A case study," in *Emerging Technologies Factory Automation (ETFA)*, 2011, pp. 1–4.

[9] I.-K. Jung and S. Lim, "An ethercat based control system for human-robot cooperation," in *Methods and Models in Automation and Robotics (MMAR)*, 2011, pp. 341–344.

[10] K. Erwinski, M. Paprocki, L. Grzesiak, K. Karwowski, and A. Wawrzak, "Application of ethernet powerlink for communication in a linux rtai open cnc system," *Industrial Electronics*, vol. 60, no. 2, pp. 628–636, 2013.

[11] (2013, Nov.) Ovda. [Online]. Available: http://www.odva.org

[12] (2013, Nov.) Profibus and profinet. [Online]. Available: http://www.profibus.com

[13] (2013, Nov.) Can in automation: Controller area network. [Online]. Available: http://www.can-cia.org

[14] P. Brooks, "Ethernet/ip-industrial protocol," in *Emerging Technologies and Factory Automation, 2001. Proceedings.*, vol. 2, 2001, pp. 505–514 vol.2.

[15] P. Ferrari, A. Flammini, D. Marioli, and A. Taroni, "Experimental evaluation of profinet performance," in *Factory Communication Systems, 2004. Proceedings.*, 2004, pp. 331–334.

[16] M. Knezic, B. Dokic, and Z. Ivanovic, "Increasing ethercat performance using frame size optimization algorithm," in *Emerging Technologies Factory Automation (ETFA)*, 2011, pp. 1–4.

[17] (2013, Nov.) Ethernet powerlink standardization group. [Online]. Available: http://www.ethernet-powerlink.org

[18] (2013, Nov.) Beaglebone black. [Online]. Available: http://beagleboard.org

[19] (2013, Nov.) Operating systems, development tools, and professional services for connected embedded systems. [Online]. Available: www.qnx.com

[20] (2013, Nov.) Xenomai: Real-time framework for linux. [Online]. Available: www.xenomai.org