OPART: Towards an Open Platform for Abstraction of Real-Time Communication in Cross-Domain Applications

Simplification of Developing Process in Real-time Networked Medical Systems

Morteza Hashemi Farzaneh*, Alois Knoll*, Jonas Pfeiffer†
*Robotics and Embedded Systems
{hashemif, knoll}@in.tum.de
†Micro Technology and Medical Device Technology
{jonas.pfeiffer}@tum.de

Technische Universität München
85748 Garching bei München, Germany

***Abstract*—Developing real-time communication in various application fields such as robotics, factory automation, etc. is one the most important steps achieving a deterministic system. However, the development of this step is very complex and requires low level and advanced knowledge about the real-time communication systems. This complexity decelerates the developing process specially in cross-domain applications e.g. surgical control applications in Networked Medical Systems (NMS) requiring real-time communication and deterministic system behavior. General complexities developing real-time communication systems are classified. The architecture of an Open Platform for Abstraction of Real-Time Communication (OPART) is introduced for reducing these complexities. The architecture of OPART is based on the Ethernet-based real-time communication protocol openPOWERLINK. An experimental setup of OPART using a medical sensor and actuator is demonstrated.**

Keywords*—Automation, Embedded systems, Industrial Ethernet, Medical, Networked Medical Systems, POWERLINK, Real-time Communication, Robotics.

I. Introduction

Real-time communication is a requirement in various application fields. However, developing such a deterministic communication requires low level, deep and advanced knowledge about hardware, real-time operating systems, real-time communication protocols and real-time programming. There are numerous real-time communication technologies [1], [2], [3], [4], [5], [6] which bring their specific properties such as specific hardware, software and infrastructural requirements. The required expert knowledge decelerates the development in cross-domain applications with real-time communication requirements. For example, surgical robotics in Networked Medical Systems (NMS) [7] requires deterministic system behavior. Common used communication systems in NMS are e.g. High Definition Multimedia Interface (HDMI) for sending and receiving high quality video data, Transmission Control Protocol/Internet Protocol (TCP/IP) [8] for exchanging non-real-time data between medical systems and IT systems such as Digital Imaging and Communications in Medicine (DICOM) [9] and Picture Archiving and Communication System (PACS) [10]. Experienced developers who are familiar with these protocols, do not necessarily have the required real-time developing background from e.g automation factory or robotics.

With this motivation, the objective of this paper is to reduce the complexity of developing cross-domain applications requiring real-time communication. The Open Platform for Abstraction of Real-Time Communication (OPART) is proposed to approach the mentioned challenges.

The paper is structured as follows: In the next section, the complexities developing real-time communication systems are defined and discussed and classified. In section III, Ethernet openPOWERLINK is explained which is the representative basis real-time communication protocol in the OPART architecture. The architecture of OPART and its implementation are described in section IV. An experimental setup for demonstration of the OPART is introduced in section V. Finally, the conclusion and future work are discussed in VI.

II. Complexities developing real-time communication systems

In this section, the complexities developing real-time communication systems are defined and classified. The OPART's components are designed to approach these complexities.

1) Real-time Node development: Each node in a real-time communication system has to fulfill timing requirements. Considering the ISO/OSI layer architecture, the first important component is the hardware which is the basis for all other components at the higher layers. Before using a hardware in a real-time communication system as a real-time node, the computational capabilities of the hardware such as Central Processing Unit (CPU) and available Random Access Memory (RAM) have to be evaluated. Also important are the number of available peripheral interfaces such as SPI, UART, I2C, etc. These interfaces are required to receive and send data from and to the application devices e.g. sensors, manipulators and robots. To reduce the the hardware-related complexity,

minimum requirements have to be defined which are used to select a hardware as a real-time communication node.
On the top of the hardware, a Real-Time Operating System (RTOS) is required that guarantees real-time scheduling of application and communication processes. Selection one of the available RTOS is another complex process developing real-time communication systems. The benchmarking factors for a RTOS are e.g. performance regarding minimum latency and jitter, costs and available drivers and the support of specific real-time communication stacks.
Running on the top of RTOS, there are various real-time communication protocol stacks which have to be evaluated. The evaluation of modern real-time communication systems including Industrial Ethernet is discussed in [7]. The evaluation and selection of such protocols is a very complex process. Hence, benchmarking and evaluation complexities have to hidden from the application developers. The results of a initial comparison shows that Ethernet Powerlink [6] as a representative technology for Industrial Ethernet is one of the suitable protocols that can be used as fundamental component of an open real-time communication system. In the next section, this technology is explained in more details.
The real-time application is placed on the top of all mentioned layers. The application layer of a real-time node is responsible to receive and send application data from and to the real-time communication system. The complexity here is in data exchange process between the application layer and the real-time communication stack. The selected method (e.g. shared memory) here does play a significant role for the final performance of real-time node.

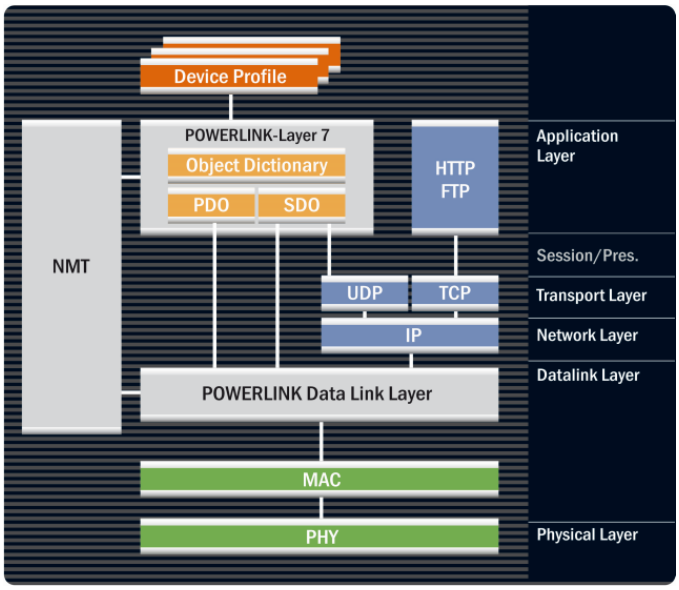

Fig. 1. Overall architecture of the openPOWERLINK real-time communication protocol [6].

2) Real-time network infrastructure: Developing a real-time communication system also depends on the network topology and required infrastructural network components. Different Network topologies lead to different network setup costs and performance. For instance, using the widespread Ethernet-based hardware in the network decreases infrastructural cost and further extensibility in comparison to other specific network components.

3) Real-time network management: A real-time communication system has to be configured and initialized in order to have a deterministic behavior. This part is one of the most complex parts because configuring such a network requires detailed knowledge about the used real-time communication stack as mentioned before. One of the main objectives of OPART is to reduce this complexity.
One limitation of real-time communication systems is the very static behavior of the system after initialization. This static behavior guarantees the real-time communication but limits the dynamic data exchanges at the run-time. For example, one sensor sends its data to two actuators in the network. This behavior is configured in the initialization phase. It can be imagined, that at the run-time an interested third actuator joins the network and wants to have the sensor data. A less suitable solution here is to stop the network, reconfigure it as desired and again start it. Reducing this complexity is one of the objectives of OPART.

III. Ethernet openPOWERLINK technology

The openPOWERLINK is an open source Industrial Ethernet technology. [6]. It is based on the standard Ethernet hardware. The architecture of the openPOWERLINK is based on the Master-Slave concept. In such an architecture, the Master synchronizes the communication between network nodes (Slaves) and guarantees real-time communication between them. In a openPOWERLINK network, the Master is called Managing Node (MN) and the Slaves are called Controlled Nodes (CN). The architecture of openPOWERLINK is presented in Figure 1. The Data Link Layer solves the problem of collisions so that each CN is only allowed to send when it gets a permission from MN.
Non-real-time data are sent through the (Internet Protocol) IP layer. The real-time data (Process Data Objects) are sent directly to the application layer. This increases the protocol performance considering response time of real-time nodes. For each openPOWERLINK network, a cycle time has is specified. A cycle time is a time period where MN sends requests (PReq packets) to all present CNs in the network and receives responses (PRes packets) from them. Figure 2 depicts the exchanged messages in a cycle. The openPOWERLINK messages are encapsulate in the standard Ethernet packets. An openPOWERLINK data packet includes a header which contains information about sender and receiver address in

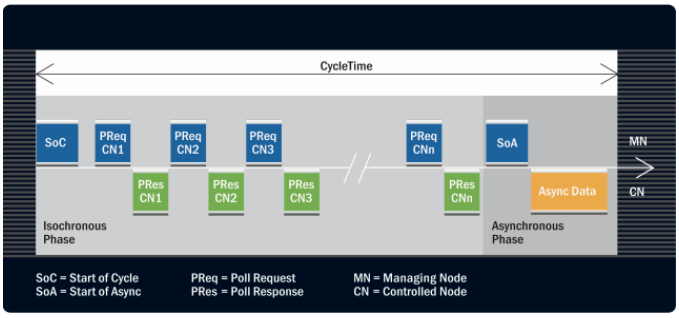

Fig. 2. At the beginning of the cycle MN sends SoC message to all CNs and then it asks all of the CNs sending PReq messages to them. CNs response with PRes packets [6].

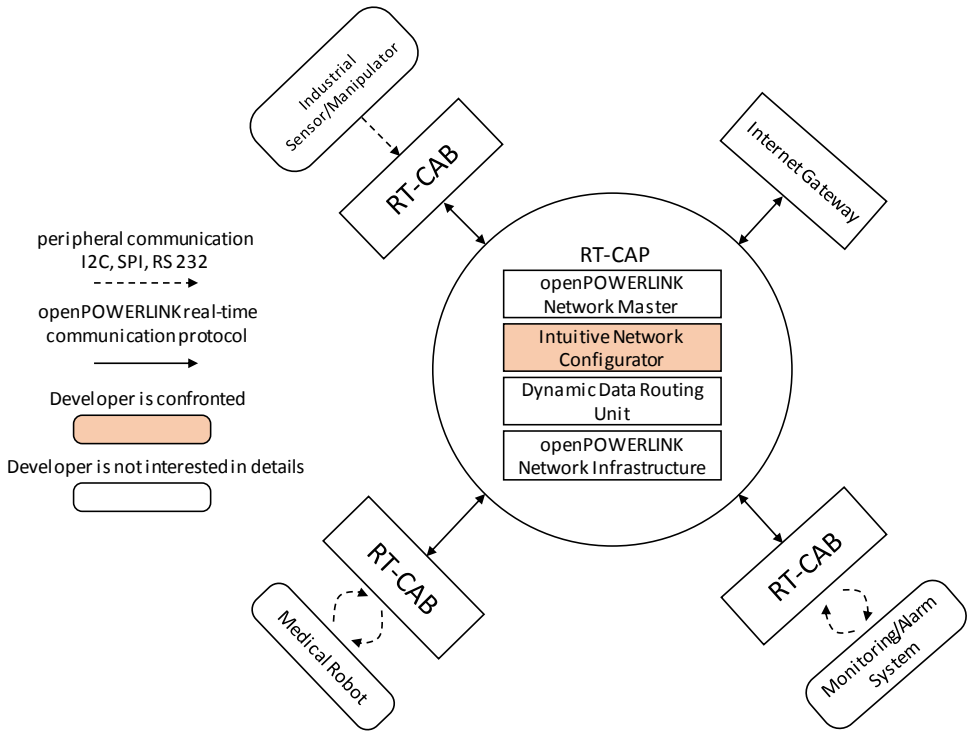

Fig. 4. OPART architecture including the Real-time Communication Abstraction Provider (RT-CAP) and Bridge (RT-CAB).

the range of 1 to 239 real-time nodes. Figure 3 shows the openPOWERLINK packet structure.
A comparison of the real-time communication systems is given in [7]. For the sake of completeness we mention the major advantages of openPOWERLINIK. Modularity and openness make openPOWERLINK to a suitable protocol which offers high flexibility in porting this protocol to different hardware platforms from very small embedded systems to desktop and industrial computers with multicore-CPUs and multiple peripheral interfaces. It also offers hot-plug-ability which leads to high reliable systems. It is based on the standard Ethernet hardware and therefore will be highly extensible in future.

IV. Architecture of OPART

Based on the classified complexities, the architectural components of OPART are introduced which approach these complexities.

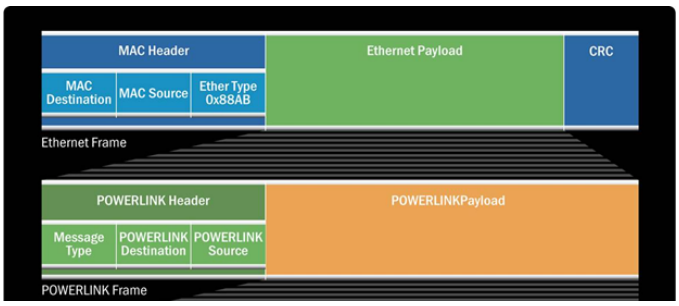

Fig. 3. The openPOWERLINK data packet structure [6].

A. Real-time Communication Abstraction Provider

Real-time Communication Abstraction Provider (RT-CAP) is the central component in the architecture. RT-CAP component covers four main sub architectural components (Figure 4). The openPOWERLINK Network Master is responsible for synchronization of the real-time nodes in OPART. The Intuitive Network Configurator is an additional component to the initial available configuration tools of openPOWERLINK (openConfigurator and Automation Studio). This component makes it possible to change the network configuration in initialization and run-time phase. The available initial configuration tools allow network modification only at the setup phase and not at the run-time. That means for each network modification, the whole network has to be stopped. Another issue with the classical configuration tools is the fact, that these modifications require deep and advanced knowledge about the used real-time communication protocol (in this case openPOWERLINK). The Intuitive Network Configurator hides these complexities. It also has to be logically separated from the specific used real-time communication protocol so that when openPOWERLINK is replaced by another protocols, the Intuitive Network Configurator does not has to be modified.
Dynamic Data Routing Unit allows to have a dynamic network behavior at the run-time. In combination with Intuitive Network Configurator and openPOWERLINK Network Master, Dynamic Data Routing Unit modifies the communication partners in the network. This sub component also has to be independent from the specific used real-time communication protocol in case of protocol changes.
Network infrastructural complexities are handled by the openPOWERLINK Network Infrastructure component. It supports the network hardware components such as HUBs and Ethernet cables. It also provides flexible network topologies which

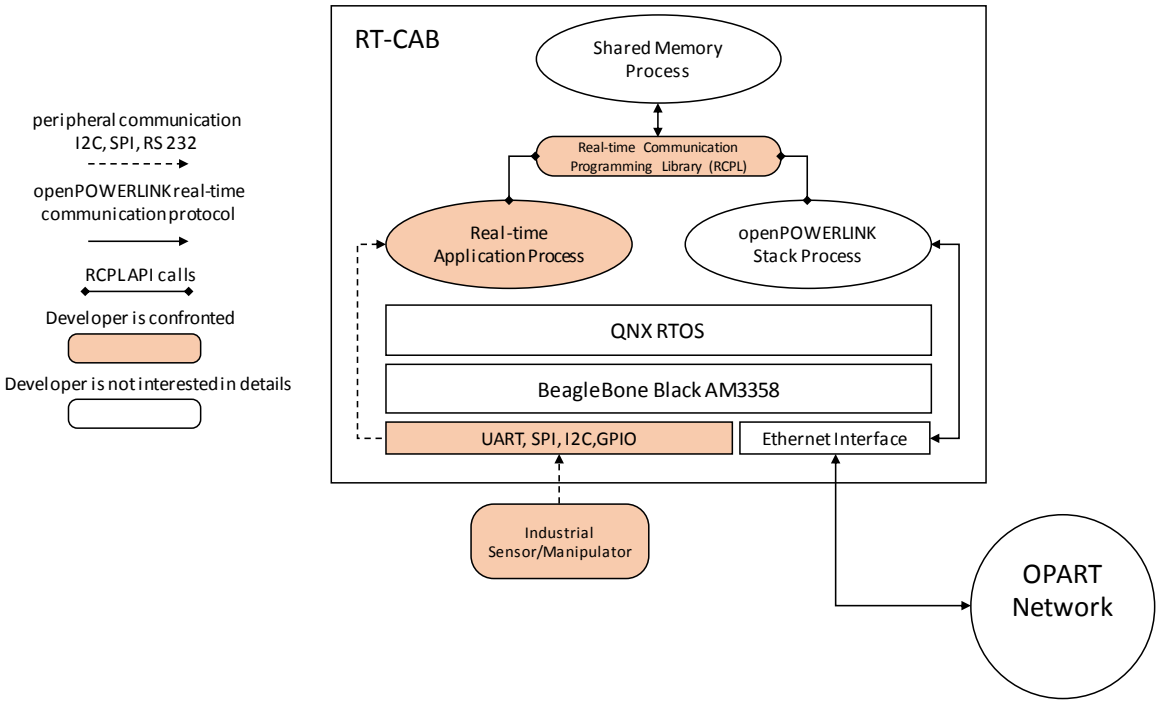

Fig. 5. Real-time Communication Abstraction Bridge (RT-CAB). Developers only have to deal with the application and the real-time network complexity is hidden by OPART RT-CAB components.

depends on the developer's and user's requirements. This depends on the selected real-time communication protocol and has to be modified if the used protocol is replaced by another one.

B. Real-time Communication Abstraction Bridge

Real-time Communication Abstraction Bridge (RT-CAB) is the main component of each real-time node in OPART. The hardware is BeagleBone Black [11] which has numerous peripheral interfaces such as UART, SPI, I2C, etc. and a powerful ARM processor AM3358 (Texas Instrument). The openPOWERLINK Slave Stack runs on top of QNX RTOS [12]. Network devices such as sensors, manipulators, robots, monitoring systems and etc. are connected to the peripheral interfaces of the RT-CAB.

There are three main processes in RT-CAB. Real-time Application Process is responsible for reading and writing from and to the peripheral interfaces and contains the application logic. It uses the Real-time Communication Programming Library (RCPL) APIs and communicates with the openPOWERLINK Stack process. The shared memory mechanism is used for the communication between application process and openPOWERLINK process. This mechanism is implemented by RCPL. RT-CAB uses the openPOWERLINK process and the related Ethernet interface for communication with the other real-time nodes of the network. Developers only have to deal with the Real-time Application Process (including the peripheral interfaces) and RCPL. The rest is provided by RT-CAB and the complexities related to the RTOS and openPOWERLINK are hidden from the developers. Figure 5 shows the subcomponents of the RT-CAB.

V. Experimental Setup

Figure 6 demonstrates an experimental and initial setup for evaluation of the proposed architecture. A surgical foot pedal (5) is used as sensor device that has to control the speed of an aspirator (1). The foot pedal is connected to its RT-CAB (3) and sends analog values to it using an I2C interface. RT-CABs are implemented as described before using the BeagleBone Black [11] embedded board and, QNX operating system and openPOWERLINK stack.

RT-CAB (3) and (2) are connected to each other using a line network topology with standard Ethernet components (HUBs and cables). RT-CAB (2) receives the analog data and forwards it to the aspirator. The X20CP1585 POWERLINK Master (4) synchronizes the communication and guaranties the real-time behavior of the network. The network is configured and the configuration is sent to it.

The Intuitive Network Configurator is the implementation phase and will extend the standard network configurators in future for a dynamic network modification at the run-time.

VI. Conclusion and Future Work

The complexities developing real-time communication systems have been discussed. Based on these complexities, the architecture of an open platform for abstraction of real-time communication (OPART) have been introduced. In OPART's architecture, openPOWERLINK technology is the used real-time communication protocol as an representative technology for Industrial Ethernet. An initial experimental setup and implementation of OPART components have been demonstrated using sensors and actuators from the Networked Medical Systems domain.

Future work will focus on further developing of the discussed

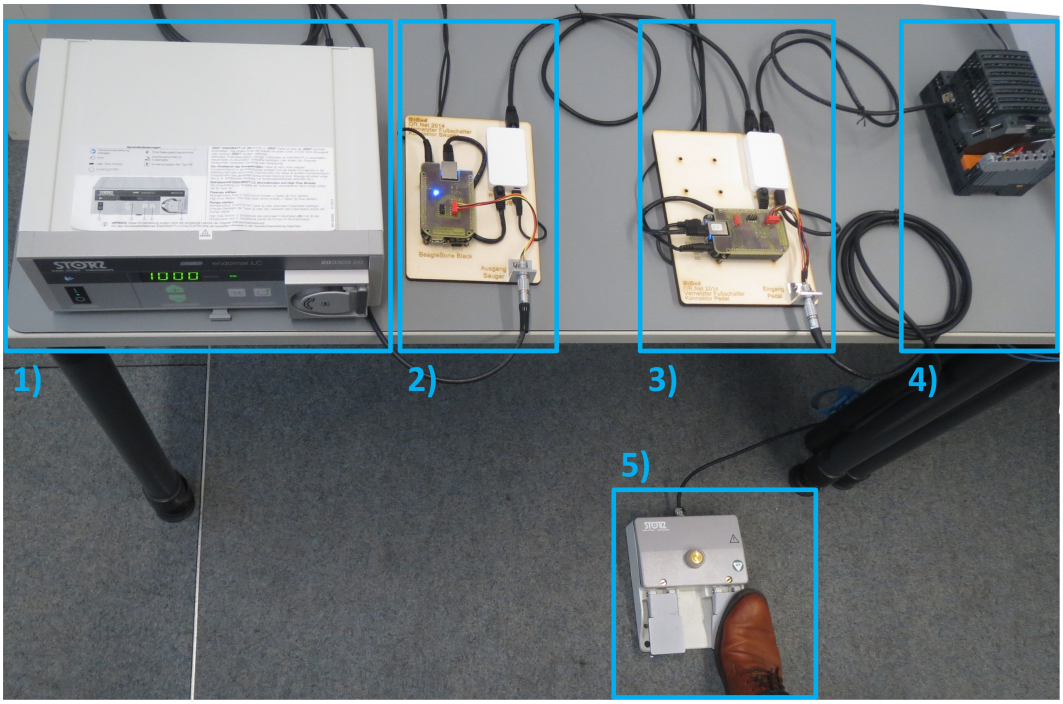

Fig. 6. Experimental setup for demonstration of OPART concept. A medical foot pedal (5) controls an aspirator (1) using the OPART components RT-CAP (4) and RT-CAB (2,3) in a line network topology.

Intuitive Network Configurator and Dynamic Data Routing Unit for enhancing the dynamic behavior of OPART at the run-time. A future work will also focus on the performance evaluation and verification of the RT-CAB.

ACKNOWLEDGMENT

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

REFERENCES

[1] (2014, Nov.) Can in automation: Controller area network. [Online]. Available: http://www.can-cia.org

[2] (2014, Nov.) Profibus and profinet. [Online]. Available: http://www.profibus.com

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

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

[5] 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.

[6] (2014, Nov.) Ethernet powerlink standardization group. [Online]. Available: http://www.ethernet-powerlink.org

[7] M. Hashemi Farzaneh, S. Nair, M. Nasseri, and A. Knoll, "Reducing communication-related complexity in heterogeneous networked medical systems considering non-functional requirements," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, Feb 2014, pp. 547–552.

[8] 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.

[9] 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. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=673987

[10] F. Breu, S. Guggenbichler, and J. Wollmann, "PACS: Personal Access Communications System - A Tutorial," *Vasa*, no. June, pp. 32–43, 2008. [Online]. Available: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf

[11] (2014, Nov.) Beaglebone black. [Online]. Available: http://beagleboard.org

[12] (2014, Nov.) Operating systems, development tools, and professional services for connected embedded systems. [Online]. Available: www.qnx.com