INTERNET-BASED EMBEDDED SYSTEM ARCHITECTURES
End-User Development Support for Embedded System Applications

Miroslav Sveda
Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic

Radimir Vrba
Faculty of Electrical Engineering & Communication, Brno University of Technology, Brno, Czech Republic

Keywords: Smart sensors, IEEE 1451, Internet, Publish/subscribe messaging, IP multicast.

Abstract: This paper presents an approach to industrial embedded system networking that offers a reusable design pattern for the class of Internet-based applications. It deals with an integrated networking framework stemming (1) from the IEEE 1451.1 smart transducer interface standard, which is an object-based networking model supporting among others publish-subscribe approach to group messaging, and (2) from the Internet Protocol (IP) multicast communication, mediating efficient and unified access to smart sensors through Internet. The kernel of this paper focuses on adaptations and tuning of those concepts and on their utilization for a gas pipes pressure measurement system as an application example. Furthermore, the contribution brings this scheme in form suitable not only for framework builders, but also for end-user developers.

1 INTRODUCTION

The design framework, presented in this paper as a flexible design environment stemming from meta-design conception, is rooted in the IEEE 1451.1 standard specifying smart transducer interface architecture. This framework enables to unify not only interconnecting smart sensors with various Fieldbuses but also their direct coupling to Ethernet-based Intranets, which are currently replacing various special-purpose Field busses in industrial applications (Sveda, et al., 2005). That standard provides an object-oriented information model targeting software-based, network independent, transducer application environments.

Two additional technologies, namely publisher-subscriber messaging and IP multicasting, which offer scalable and traffic-saving solution important in the context of contemporary Internet, complement the framework providing design patterns reusable for various sensor-based networked systems. The schemes discussed can properly interplay with each other and can provide suitable support for Internet-based sensor systems design. The kernel of this paper focuses on adaptations and tuning of introduced concepts and on their utilization for a network-based pressure measurement system as a real-world project. This pilot application implements a distributed, gas-pipe's measurement arrangement. It comprises groups of smart pressure and temperature sensors that clients can access effectively through Internet. Each sensor group is supported by an active web page with Java applets that, after downloading, provide clients with transparent and efficient access to pressure measurement services over such geographically distributed objects as the large-scale systems of gas pipes or similar industrial applications.

The paper discusses in the following section meta-design basic principles utilizable for creating flexible design environment reusable for various application domains of sensor systems. Next three sections introduce subsequently IEEE 1451 package of communication standards, client-server and publisher-subscriber communication concepts, and Internet Protocol (IP) multicasting as main abstract components of the design pattern forming the heart of the generic development environment. The section 6 presents in more detail an example of employment of this framework for designing networked distributed embedded systems for pressure measurement along gas pipes. The example covers design of network configuration, implementation concepts developed, node configuration, and smart sensor implementation.

2 META-DESIGN

Component-based development involves multiple roles (Morch, et al., 2004). Framework builders create the infrastructure for components to interact; developers identify suitable domains and develop new components for them; application assemblers select domain-specific components and assemble them into applications; and end users employ component-based applications to perform daily tasks. There is room for a fifth role in this pipe-line: end-user developers positioned between application assemblers and end users. These end-user developers are able to tailor applications at runtime because they have both domain expertise and technical know-how. They would interact with applications to adjust individual components, and modify existing assemblies of components to create new functionality. Furthermore, they can play a critical role when component-based systems have to be redesigned for new requirements. End-user development activities can range from customization to component configuration and programming.

Meta-design characterizes objectives, techniques, and processes for creating new environments allowing end users to act as designers (Fischer, et al., 2004). In all design processes, two basic stages can be differentiated: design time and use time. At design time, system developers create environments and tools. In conventional design they create complete systems. Because the needs, objectives, and situational contexts of users can only be anticipated at design time, users often find the system unfit for their tasks at use time. Thus, they require adaptation of the existing environment and tools for new applications. Meta-design extends the traditional notion of system development to include users in an ongoing process as co-designers, not only at design time but throughout the entire life-cycle of the development system. Rather than presenting users with closed systems, meta-design provides them with concepts and tools to extend the system to fit their needs. Hence, meta-design promotes designing the design process.

This paper discusses a deployment of meta-design principles for creating a flexible design environment focused on sensor systems interconnected by Internet aiming namely at industrial applications. Necessarily under-designed open source tools and techniques create design spaces for end-user developers in such application domains as pressure measurement along gas pipes, which is used as a case study demonstrating the principles of such environment exploitation.

3 IEEE 1451 SET OF STANDARDS

The IEEE 1451 package consists of the family of standards for a networked smart transducer interface that include namely (see Figure 1) (i) a smart transducer software architecture, 1451.1 (IEEE, 2000), targeting software-based, network independent, transducer applications, and (ii) a standard digital interface and communication protocol, IEEE 1451.2, for accessing the transducer or the group of transducers via a microprocessor modeled by the 1451.1 standard. The next three standard proposals extend the original hard-wired parallel interface 1451.2 to serial multi-drop 1451.3, mixed-mode (i.e. both digital and analogue) 1451.4, and wireless 1451.5 interfaces.

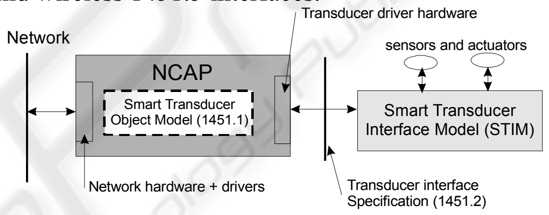
Transducer driver hardware
Network
sensors and actuators
NCAP
Smart Transducer Object Model (1451.1)
Smart Transducer Interface Model (STIM)
Network hardware + drivers
Transducer interface Specification (1451.2)

Figure 1: IEEE 1451 configuration example.

The 1451.1 software architecture provides three models of the transducer device environment: (i) the object model of a network capable application processor (NCAP), which is the object-oriented embodiment of a smart networked device; (ii) the data model, which specifies information encoding rules for transmitting information across both local and remote object interfaces; and (iii) the network communication model, which supports client/server and publish/subscribe paradigms for communicating information between NCAPs. The standard defines a network and transducer hardware neutral environment in which a concrete sensor/actuator application can be developed.

The object model definition encompasses the set of object classes, attributes, methods, and behaviors that specify a transducer and a network environment to which it may connect. This model uses block and base classes offering patterns for one Physical Block, one or more Transducer Blocks, Function Blocks, and Network Blocks. Each block class may include specific base classes from the model. The base classes include Parameters, Actions, Events, and Files, and provide component classes.

All classes in the model have an abstract or root class from which they are derived. This abstract class includes several attributes and methods that are common to all classes in the model and provide a definition facility for the instantiation and deletion of concrete classes including attributes.

Block classes form the major blocks of functionality that can be plugged into an abstract card-cage to create various types of devices. One Physical Block is mandatory as it defines the card-cage and abstracts the hardware and software resources that are used by the device. All other block and base classes can be referenced from the Physical Block.

The Transducer Block abstracts all the capabilities of each transducer that is physically connected to the NCAP I/O system. During the device configuration phase, the description is read from the hardware device what kind of sensors and actuators are connected to the system. The Transducer Block includes an I/O device driver style interface for communication with the hardware. The I/O interface includes methods for reading and writing to the transducer from the application-based Function Block using a standardized interface. The I/O device driver provides both plug-and-play capability and hot-swap feature for transducers.

The Function Block provides a skeletal area in which to place application-specific code. The interface does not specify any restrictions on how an application is developed. In addition to a State variable that all block classes maintain, the Function Block contains several lists of parameters that are typically used to access network-visible data or to make internal data available remotely.

The Network Block abstracts all access to a network employing network-neutral, object-based programming interface supporting both client-server and publisher-subscriber patterns for configuration and data distribution.

4 CLIENT-SERVER AND PUBLISHER-SUBSCRIBER PATTERNS

The majority of communication protocols provide a client-server style of communication. In case of sensor communications, the client-server pattern covers both configuration of transducers and initialization actions. If the client wants to call some function on server side, it uses a command *execute*. On server side, this request is decoded and used by the function *perform*. That function evaluates the requested function with the given arguments and, after that, it returns the resulting values to the client.

The client-server pattern corresponds to remote procedure call (RPC), which is the remote invocation of operations in a distributed context (Eugster, et al., 2003). To be more precise, the RPC interaction considered in this paper provides a synchronous client-server communication, i.e. the client is waiting for a server's response before completion the RPC actions related to the current call. Evidently, the client-server communication style relates to point-to-point message passing called as unicast.

The subscriber-publisher style of communication (Eugster, et al., 2003) can provide the efficient distribution of measured data. All clients, wishing to receive messages from a transducer, register themselves to the group of its subscribers using the function *subscribe*. After that, when this transducer generates a message using the function *publish*, this message is effectively delivered to all members of its subscribing group. Transducers in the role of publishers have also the ability to advertise the nature of their future events through an *advertise* function.

The interaction publish-subscribe relates to point-to-multipoint or multipoint-to-multipoint asynchronous message passing. Of course, it can be implemented using multiple unicast communication transactions. On the other hand, to satisfy the requirement of efficiency, it is necessary to utilize elaborate multicast techniques encompassing multicast addressing and, namely, multicast routing. The basic principles of the network layer multicast in the Internet environment are discussed in the following section.

5 MULTICASTING

Traditional network computing paradigm involves communication between two network nodes. However, emerging Internet applications require simultaneous group communication based on multipoint configuration propped e.g. by multicast IP, which saves bandwidth by forcing the network to replicate packets only when necessary. Multicast improves the efficiency of multipoint data distribution by building distribution trees from senders to sets of receivers (Miller, 1999).

The functions that provide the Standard Internet Multicast Service can be separated into host and network components. The interface between these components is provided by IP multicast addressing and Internet Group Management Protocol (IGMP) group membership functions, as well as standard IP packet transmission and reception. The network functions are principally concerned with multicast routing, while host functions can also include higher-layer tasks such as the addition of reliability facilities in a transport-layer protocol.

IP multicasting is the transmission of an IP datagram to a host group, a set of zero or more hosts identified by the single IP destination address of class D. Multicast groups are maintained by IGMP (IETF RFC 1112, RFC 2236). Multicast routing considers multicasting routers equipped with multicast routing protocols such as DVMRP (RFC 1075), MOSPF (RFC 1584), CBT (RFC 2189), PIM-DM (RFC 2117), PIM-SM (RFC 2362), or MBGP (RFC 2283). For Ethernet-based Intranets, the Address Resolution Protocol provides the last-hop routing by mapping class D addresses on multicast Ethernet addresses.

6 CASE STUDY

The presented case study, used to demonstrate the introduced concepts, includes several groups of smart pressure and temperature sensors that clients can access effectively through Internet. Each sensors group is supported by an active web page with Java applets that, after downloading, provide clients with transparent and efficient access to pressure measurement services over such geographically distributed objects as the considered large systems of gas pipes. The complete system comprises several groups of smart pressure sensors complemented by temperature sensors that enable computing of temperature corrections (Sveda and Vrba, 2003).

6.1 Network Configuration

Each sensor group is supported by an active web page with Java applets that, after downloading, provide clients with transparent and efficient access to pressure measurement services. This section demonstrates the above-introduced concepts and tools adapted and applied to the development of such a gas-pipes pressure analyzer

In this case, clients communicate to transducers using a messaging protocol defined by client-server and subscriber-publisher patterns employing 1451.1 Network Block functions. A typical configuration includes a set of smart pressure sensors generating pressure values for the users of those values. To register itself for a specified group of sensors, the user — playing the role of either subscriber or client — opens the related server's web page with the relevant Java applet. This applet is, after uploading to the subscriber/client site, started on subscriber/client's computer, which launches communications with a group of transducers allowing Java clients to connect and subscribe to the smart sensors. Java can directly support both client-server and subscriber-publisher application architectures as the core Java specifications include TCP/IP and UDP/IP networking APIs.

The developed Java applet uses the core java.net package to implement both client-server and subscriber-publisher application distribution allowing to access smart sensors and supporting nodes. The applet consists of a series of object classes, including multi-threaded applet environment, animation, and UDP/IP-based subscriber and TCP/IP-based client communications. The subscriber/client software implemented in Java enables applets to be included in a web server HTML page, and run under a regular web browser on subscriber/client side. The subscriber/client communicates with the transducer by standard UDP/TCP sockets employing IP multicast.

The communication scheme applies multicast both for distributing measured values from a transducer to a group of subscribers/clients registered by the web server for this transducer, and for spreading commands of a client to a group of transducers registered for this client.

6.2 Implementation Concepts

In the transducer's 1451.1 object model, basic Network Block functions initialize and cover communication between a client and the transducer, which are identified by unique unicast IP addresses. The client-server style communication, which in this application covers both the configurations of transducers and initialization actions, is provided by two basic Network Block functions: *execute* and *perform*. The standard defines a unique ID for every function and data item of each class. If the client wants to call some function on server side, it uses command *execute* with the following parameters: ID of requested function, enumerated arguments, and requested variables. On server side, this request is decoded and used by the function *perform*. That function evaluates the requested function with the given arguments and, in addition, it returns the resulting values to the client. Those data are delivered by requested variables in *execute* arguments.

The subscriber-publisher style of communication, which in this application covers primarily distribution of measured data, but also distribution of group configuration commands, employs IP multicasting. All clients wishing to receive messages from a transducer, which is joined with an IP multicast address of class D, register themselves to this group using IGMP. After that, when this

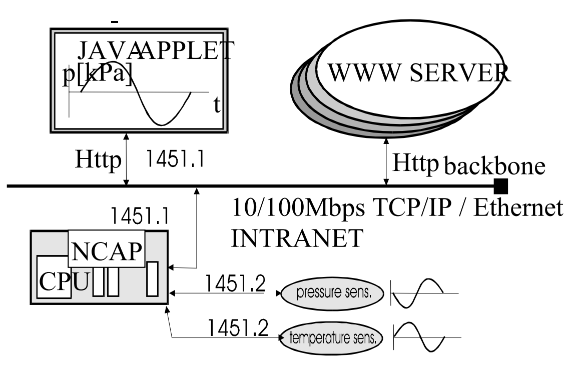

Figure 2: Sensor node configuration example.

transducer generates a message by Block function *publish*, this message is effectively delivered to all members of this class D group, without unnecessary replications and repeated transmissions.

The Network Block abstracts all access to a network employing network-neutral, object-based programming interface. The network model provides an application interaction mechanism supporting both client-server and publisher-subscriber paradigms for event and message generation and distribution.

6.3 Node Configuration

The primary communication scheme, which is based on publish-subscribe application interface, applies multicast both for distributing measured values from a transducer to a group of clients registered by the www server for this transducer, and for spreading commands of a client to a group of transducers registered for this client. Those commands can specify e.g. individual subgroup's sampling frequencies and/or events for launching irregular publishing such as a limit value crossing.

A typical node, depicted on Figures 2 and 3, consists of STIM (Smart Transducer Interface Module) connected with PSD sensor for pressure measurements, and with auxiliary temperature sensor for signal conditioning. Of course, NCAP can be either embedded in a complex smart sensor, or shared among more simple smart sensors. On the other hand, from the viewpoint of Internet, only NCAP is directly addressable being equipped by its own IP address. Therefore, we can also denote as smart sensor the device consisting of an NCAP accessing one or more STIMs with connected sensors. To register itself for a specified group of sensors, the client opens a related server's web page with the relevant Java applet. This applet is, after uploading to the client site, started on client's computer, what launches communication with the dedicated group of transducers.

6.4 Smart Sensor Implementation

This subsection discusses, as an example, the pressure sensors with reflected laser beam and diffractive lens. The sensitive pressure sensor is based on a nitride membrane and an optoelectronic read-out subsystem. Measured pressure values are transformed into related thick-layer nitride membrane deflections. The nitride membrane serves as a mirror for laser beam, and it can move the related reflected laser mark. The mark's position is sensed using position-sensing device, which is a fotolateral diode. Diode double current signal is amplified and conditioned digitally by the ADuC812 microcontroller. This single-chip microcontroller provides also the IEEE 1451.2 interface.

The sensing subsystem combines two principles that provide both high precision and wide range pressure measurements. Large displacements are measured by the position of reflected focused laser beam. Small position changes are measured by one-side layer diffractive lens principle. Sensor output signal is conditioned in digital by the ADuC812 single-chip microcontroller, which provides the

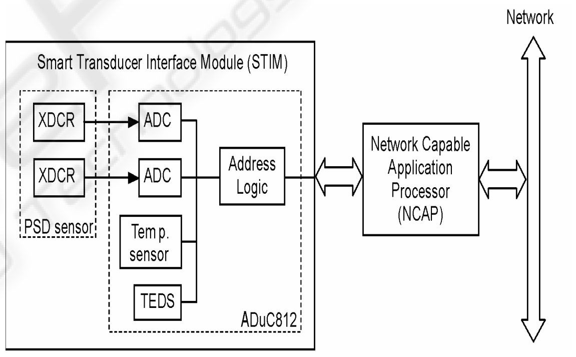
Network
Smart Transducer Interface Module (STIM)
XDCR
XDCR
PSD sensor
ADC
ADC
Address Logic
Temp. sensor
TEDS
ADuC812
Network Capable Application Processor (NCAP)

Figure 3: Sensor node implementation example.

IEEE1451.2 interface as one of its communication ports. This microcontroller calculates the position of the light spot and converts that position on the measured pressure using an internal table.

Figure 3 depicts principles of the implementation of that smart sensor. The STIM contains (1) a PSD sensor with two analog differential transducers (XDCR), (2) a microcontroller ADuC812 with nonvolatile memory containing a TEDS field (Transducer Electronic Data Sheet) that props IEEE 1451.2 storing sensor specifications, (3) a TII (Transducer Independent Interface), (4) a temperature sensor necessary for signal conditioning, (5) an analogue-to-digital conversion units (ADC), and (6) a logic circuitry to facilitate communication between the STIM and the NCAP.

The ADuC812 microcontroller, the basic building block of the smart pressure sensor electronics, includes on-chip high performance multiplexers, ADCs, DACs, FLASH program and data storage memory, an industrial standard 8052 microcontroller core, and supports several standard serial ports. The microcontroller may also utilize nonvolatile memory containing a TEDS field and ten-wire TII that prop IEEE 1451.2.

7 CONCLUSIONS

This contribution deals with industrial, sensor-based applications development support aiming at distributed components interconnected by Internet compatible intranets. The paper presents an approach to embedded system networking that offers a reusable design pattern for a class of Internet-based applications. It aims at an integrated networking framework stemming (1) from the IEEE 1451.1 smart transducer interface standard, which is an object-based networking model supporting client-server and publish-subscribe communication patterns in group messaging, and (2) from the IP multicast communication, mediating efficient access to smart sensors through Internet.

The pilot application demonstrates that clients can access groups of smart pressure and temperature sensors effectively through Internet. Each sensors group can be supported by an active web page with Java applets that, after downloading, can provide clients/subscribers with transparent and efficient access to measurement services flexible enough to satisfy various application requirements.

The newly established application domain of industrial sensor networks brings special requirements not only on safety and security, but also on reuse of platforms developed originally for different domains. This paper deals with a concrete industrial embedded system networking design approach that employs meta-design principles for reuse IEEE 1451.1 smart transducer object model with publish-subscribe messaging over IP multicasting, which mediate efficient access from Internet to sensors and vice versa. Such solution can offer rapid tailoring of development environments aiming, among others, at sufficient response times also in frame of unpredictable Internet background traffic.

This paper discusses a deployment of meta-design principles for creating a flexible design environment focused on sensor systems interconnected by Internet aiming namely at industrial applications. Necessarily under-designed open source tools and techniques create design spaces for end-user developers in such application domains as pressure measurement along gas pipes, which is used as a case study demonstrating the principles of such environment exploitation. The sensor-based embedded systems accessible via standard Internet and based on the IEEE 1451.1 standard as described in the paper can be simply reused, modified or redesigned to new systems not only for industrial, but also for scientific, medical, biological and other purposes.

ACKNOWLEDGEMENTS

This research has been partly funded by the Czech Ministry of Education in frame of the Research Intention No. MSM 0021630503 MIKROSYN: New Trends in Microelectronic Systems and Nanotechnologies; and by the Grant Agency of the Czech Republic through the grants GACR 102/05/0723: A Framework for Formal Specifications and Prototyping of Information System's Network Applications, and GACR 102/05/0467: Architectures of Embedded Systems Networks.

REFERENCES

Eugster, P.T., et al., 2003. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, Vol. 35, pp.114-131.

Fischer, G. et. al., 2004. Meta-Design: A Manifesto for End-User Development. *Communications of the ACM*, Vol.47, No.9, pp.33-37.

IEEE, 2000. *IEEE 1451.1, Standard for a Smart Transducer Interface for Sensors and Actuators -- Network Capable Application Processor (NCAP) Information Model*.

Miller, C.K., 1999. *Multicast Networking and Applications*, Addison-Wesley, Reading, Massachusetts, USA.

Morch, A.I., et. al., 2004. Component-Based Technologies for End-User Development. *Communications of the ACM*, Vol.47, No.9, pp.59-62.

Sveda, M. and R. Vrba, 2003. An Integrated Framework for Internet-Based Applications of Smart Sensors. *IEEE Sensors Journal,* Vol.3, No. 5, pp.579-586.

Sveda, M., et al., 2005. Introduction to Industrial Sensor Networking, A book chapter in: Ilyas, M., and I. Mahgoub, (Eds.), 2005. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, CRC Press LLC, Boca Raton, FL, USA.