TOWARDS A UNIFIED SECURITY/SAFETY FRAMEWORK
A Design Approach to 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: Embedded software, security, safety.

Abstract: This paper presents a safety and security-based approach to networked embedded system design that offers reusable design patterns for various domain-dedicated applications. After introducing proper terminology, it deals with industrial, sensor-based applications development support aiming at distributed components interconnected by wired Internet and/or wireless sensor networks. The paper presents a dependability-driven approach to embedded networks design for a class of Internet-based applications. It discusses an abstract framework stemming from embedded system networking technologies using wired and wireless LANs, and from the IEEE 1451.1 smart transducer interface standard supporting client-server and publish-subscribe communication patterns with group messaging based on IP multicast that mediate safe and secure access to smart sensors through Internet and Zigbee. The case study demonstrates how clients can access groups of wireless smart pressure and temperature sensors and safety valves through Internet effectively using developed system architecture, which respects prescribed requirements for application dependent safety and security.

1 INTRODUCTION

This paper deals with a safety and security-based approach to networked embedded system's design that is rooted in design patterns reusable for various application domains. The approach aims at embedded system design for a class of Internet-based applications focusing on a conceptual framework that stems from (1) embedded system networking technologies using wired and wireless LANs, and (2) the IEEE 1451.1 smart transducer interface specification supporting client-server and publish-subscribe communication styles. That communication employs group messaging based on IP multicast, mediating safe and secure access to smart sensors through Internet and ZigBee.

After introducing basic concepts of functionality and dependability, the paper discusses relations in between safety and security properties that embrace main functional and non-functional requirements in the considered application domain. Next section introduces embedded system networking, in this case ZigBee and IEEE 1451 protocols, client-server and publish-subscribe configurations, and multicasting including secure multicast. The case study demonstrates how clients can access groups of wireless smart pressure and temperature sensors and safety valves effectively through Internet using developed system architecture while respecting prescribed requirements for application dependent safety and security.

2 FUNCTIONALITY AND DEPENDABILITY

The design of current networked embedded system applications should consider functionality and dependability measures. Functionality means services delivery in the form and time fitting requirement specifications, where the service specification is an agreed description of the expected service. Functionality properties should be realized efficiently and cost-effectively, so reachable

Sveda M. and Vrba R. (2007).
TOWARDS A UNIFIED SECURITY/SAFETY FRAMEWORK - A Design Approach to Embedded System Applications.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 345-350
DOI: 10.5220/0001324503450350

performance and maintainability of implementation belong to the checked properties.

Dependability is that property of a system that allows reliance to be justifiably placed on the service it delivers. A failure occurs when the delivered service deviates from the specified service. Dependability measures consist of reliability, availability, security, safety and survivability. Availability is the ability to deliver shared service under given conditions for a given time, which means elimination of denial-of-service vulnerabilities. Security, see e.g. (Kim et al., 2006), is the ability to deliver service under given conditions without unauthorized disclosure or alteration of sensitive information. It includes privacy as assurances about disclosure and authenticity of senders and recipients. Security attributes add requirements to detect and avoid intentional faults. Safety, see (Leveson, 1984), is the ability to deliver service under given conditions with no catastrophic effects. Safety attributes add requirements to detect and avoid catastrophic failures.

A failure occurs when the delivered service deviates from the specified service. The failure occurred because the system was erroneous: an error is that part of the system state which is liable to lead to failure. The cause of an error is a fault. Failures can be classified according to consequences upon the environment of the system. While for benign failures the consequences are of the same order of magnitude (e.g. cost) as those of the service delivered in the absence of failure, for malign or catastrophic failures the consequences are not comparable.

A fail-safe system attempts to limit the amount of damage caused by a failure. No attempt is made to satisfy the functional specifications except where necessary to ensure safety. A mishap is an unplanned event (e.g. failure or deliberate violation of maintenance procedures) or series of events that results in damage to or loss of property or equipment. A hazard is a set of conditions within a state from which there is a path to a mishap.

A fail-stop system never performs an erroneous state transformation due to a fault (Schneider, 1983). Instead, the system halts and its state is irretrievably lost. The fail stop model, originally developed for theoretical purposes, appears as a simple and useful conception supporting the implementation of some kinds of fail-safe systems. Since any real solution can only approximate the fail-stop behavior and, moreover, the halted system offers no services for its environment, some fault-avoidance techniques must support all such implementations.

2.1 Safety-Security Relationships

Evidently, design of any safe system requires deploying security to avoid intentional catastrophic failures. And vice versa, system's security can be attacked using a safety flaw. The greater the assurance, the greater the confidence that a security system will protect against threads, with an acceptable level of risk (Kim, 2006).

The above statement deals with trust, which is assured reliance on the character, ability, strength, or truth of someone or something (Li and Singhal, 2007). Trust can be defined as the belief that an entity is capable of acting reliably, dependably, and securely in a particular case. In frame of network systems, trust is a complex subject that should be managed. Trust management entails collecting the information necessary to establish a trust relationship and dynamically monitoring and adjusting the existing trust relationship.

Principally, security represents the combination of confidentiality, integrity and availability (Kim, 2006). Obtaining assurance that the system behavior will not result in unauthorized access is called a safety problem (McLean, 1994), which is undecidable in general, see (Harrison et al., 1976), but essential to solve in concrete situations.

3 EMBEDDED SYSTEM NETWORKING

Embedded system networking stems from hierarchically interconnected networks, mostly Internet, local area wired and wireless networks, and wireless sensor networks. Internet access to individual components of distributed embedded systems can be based on both wired and wireless LAN technologies, predominantly on IEEE 802.3 and related Ethernet standards, and on IEEE 802.11b WiFi and associated wireless LAN protocols. Embedded systems and their components can be attached directly to Ethernet with TCP/IP protocol stack, but also indirectly or exclusively through various wired Fieldbuses or wireless technologies such as IEEE 802.11b WiFi and IEEE 802.15.4 with related ZigBee. Sensor networks bring an important pattern with single base station connected to a wired network on one side and wirelessly to smart sensors on the other side. When sensors are clustered, the base station communicates

to cluster heads and through them to individual sensors.

Application dedicated architectures can profit from IEEE 1451.0, .1, .2, .3, .4 and .5 standards. In this case, application layer can stem from object-oriented abstractions introduced by IEEE 1451.1 Network Capable Application Processor (NCAP) model.

3.1 ZigBee

The ZigBee/IEEE 802.15.4 protocol profile, see (ZigBee, 2004) and (IEEE 802.15.3, 2003), is intended as a specification for low-powered wireless networks. ZigBee is a published specification set of high level communication protocols designed to use small low power digital radios based on the IEEE 802.15.4 standard for wireless personal area networks. The document 802.15.4 specifies two lower layers: physical layer and medium access control sub-layer. The ZigBee Alliance builds on this foundation by providing the network layer and the framework for application layer, which includes application support sub-layer covering ZigBee device objects and manufacturer-defined application objects.

Responsibilities of the ZigBee network layer include mechanisms used to join and leave a network, to apply security to frames and to route frames to their intended destinations. In addition to discovery and maintenance of routes between devices including discovery of one-hop neighbors, it stores pertinent neighbor information. The ZigBee network layer supports star, tree and mesh topologies. Star topology network is controlled by one single device called ZigBee coordinator, which is responsible for initiating and maintaining devices on the network. Those devices, known as end devices, directly communicate with the ZigBee coordinator. In mesh and tree topologies, the ZigBee coordinator is responsible for starting the network and for choosing key network parameters.

Each network may be extended through the use of ZigBee routers. In tree networks, routers move data and control messages through the network using a hierarchical routing strategy. Tree networks may employ beacon-oriented communication as described in the IEEE 802.15.4 specification. Mesh networks allow full peer-to-peer communication. ZigBee routers in mesh networks shall not emit regular IEEE 802.15.4 beacons.

The ZigBee application layer includes application support sub-layer, ZigBee device objects and manufacturer-defined application objects. The application support sub-layer maintains tables for binding, which is the ability to match two devices together based on their services and their needs, and forwards messages between bound devices. The responsibilities of the ZigBee device objects include defining the role of the device within the network (e.g., ZigBee coordinator or end device), initiating and/or responding to binding requests and establishing a secure relationship between network devices. The ZigBee device object is also responsible for discovering devices on the network and determining which application services they provide.

3.2 IEEE 1451

The IEEE 1451 package consists of the family of standards for a networked smart transducer interface that include (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 (Sveda, 2005).

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 of what kind of sensors and actuators are connected to the system is read from the hardware device. 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 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.

3.3 Client-Server and Publisher-Subscriber Patterns

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.

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.

3.4 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 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.

3.5 Secure Multicast

In frame of a trust management in distributed systems (Li and Singhal, 2007), multicast security should provide assurance about disclosure (privacy) and authenticity of sender/recipient. The key exchange protocols used between unicast hosts do not scale well to groups. Re-keying is required to maintain confidentiality as group membership changes. The IETF Multicast Security and IRTF Group Security working groups developed a building block approach to solve the problem. The blocks encompass data security transforms, group key management and security association, and group policy management. Any application may use different blocks together to create a protocol that meets its specific requirements.

4 CASE STUDY

The case study, based on a real-world application, which was introduced in more detail but from distinct perspectives by (Sveda and Vrba, 2003) and (Sveda and Vrba, 2006), is used in this paper to demonstrate basic features of safety and security conception of the application architecture.

4.1 Application Architecture

The application architecture comprises several groups of wireless pressure and temperature sensors with safety valve controllers as base stations connected to wired intranets that dedicated clients can access effectively through Internet. The web server supports each sensor group by an active web page with Java applets that, after downloading, provide clients with transparent and efficient access to pressure and temperature measurement services through controllers. Controllers provide clients not only with secure access to measurement services over systems of gas pipes, but also communicate to each other and cooperate so that the system can resolve safety and security-critical situations by shutting off some of the valves.

Each wireless sensor group is supported by its controller providing Internet-based clients with secure and efficient access to application-related services over the associated part of gas pipes. In this case, clients communicate to controllers using a messaging protocol based on client-server and subscriber-publisher patterns employing 1451.1 Network Block functions. A typical configuration includes a set of sensors generating pressure and temperature values for the related controller that computes profiles and checks limits for users of those or derived values. When a limit is reached, the safety procedure closes valves in charge depending on safety service specifications.

In the transducer's 1451.1 object model, basic Network Block functions initialize and cover communication between a client and the transducer. The client-server communication style, which in this application covers configurations of transducers, 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 requests to call any of the functions 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.

4.2 Application Safety and Security

The application architecture comprises several groups of wireless pressure and temperature sensors with safety valve controllers as base stations connected to wired intranets that dedicated clients can access effectively through Internet, see Figure 1. The WWW server supports each sensor group by an active web page with Java applets that, after downloading, provide clients with transparent and efficient access to pressure and temperature measurement services through controllers. Controllers provide clients not only with secure access to measurement services over systems of gas pipes, but also communicate to each other and cooperate so that the system can resolve safety and security-critical situations by shutting off some of the valves.

Each wireless sensor group is supported by its controller providing Internet-based clients with secure and efficient access to application-related services over the associated part of gas pipes. In this case, clients communicate to controllers using a messaging protocol based on client-server and subscriber-publisher patterns employing 1451.1 Network Block functions. A typical configuration includes a set of sensors generating pressure and temperature values for the related controller that computes profiles and checks limits for users of those or derived values. When a limit is reached, the safety procedure, which is derived from the fail-stop model discussed in section 2, closes valves in charge depending on safety service specifications.

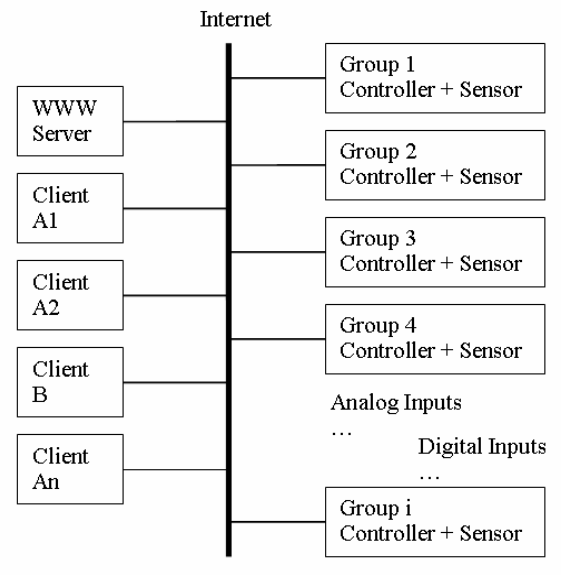

Figure 1: Network configuration example.

Security in frame of Intranet subnets can be based on current virtual private network concepts. The discussed application utilizes ciphered channels based on tunneling between a client and a group of safety valve controllers. The tunnels are created with the support of associated authentications of each client.

5 CONCLUSIONS

The paper presents an approach to embedded system networking that offers a reusable design pattern for the considered class of Internet-based applications. It brings an integrated networking framework stemming from the IEEE 1451.1 smart transducer interface standard, which represents an object-based networking model supporting client-server and publish-subscribe communication patterns in group messaging, and from the IP multicast communication, which mediates safe and secure access to sensors and actuators in sensor networks through Internet.

ACKNOWLEDGEMENTS

The research has been supported by the Czech Ministry of Education in the frame of Research Intentions MSM 0021630528: Security-Oriented Research in Information Technology and MSM 0021630503 MIKROSYN: New Trends in Microelectronic Systems and Nanotechnologies; and in part 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.

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

IEEE 802.15.4, 2003. *Wireless Medium Access Control and Physical Layer Specification for Low-Rate Wireless Personal Area Networks*, IEEE, New York, USA.

Kim, I.-G., et al., 2006. Formal Verification of Security Model using SPR Tool. *Computeing and Informatics*, Vol.25, No.5, pp.353-368.

Leveson, N.G., 1984. Software Safety in Computer-Controlled Systems. *IEEE Computer*, Vol.17, No.2, pp. 48-55.

Li, H. and M. Singhal, 2007. Trust Management in Distributed Systems. *IEEE Computer*, Vol.40, No.2, pp. 45-53.

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

Schneider, F.B., 1983. Fail-Stop Processors. *Digest of Papers 26th IEEE CS Int. Conf. COMPCON'83 SPRING*, pp. 66-70.

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.

Sveda M. and R. Vrba, 2006. Internet-Based Embedded System Architectures -- End-User Development Support for Embedded System Applications. *Proceedings of the International Joint Conference on e-Business and Telecommunications (ICETE 2006, ICE-B)*, INSTICC and IEEE, Setúbal, Portugal, 2006, pp.63-68.

ZigBee Alliance, 2004. ZigBee Specification v 1.0. ZigBee Alliance Board of Directors, Website http://www.zigbee.org/