Model based code generation approach for fast-deployment wireless security applications

László Szabados, András G. Tóth, Gyula Simon

University of Pannonia, Department of Computer Science
H-8200, Veszprém, Egyetem u. 10, Hungary
simon@dcs.vein.hu

Abstract: A diverse set of sensors and actuators are key components of integrated security systems, which provide protection against various types of attacks and threats. Based on the type of the protected objects and environment the sensor/actuator components can be completely different, and the control logic, which makes decision based on the sensor readings, must be configured to the actual scenario. Short-lifetime security systems require fast and cost effective deployment but the safety requirements are still high. In this paper an architecture and a corresponding model-based code generation scheme is proposed, which provides easy and fast deployment for security applications with various sensory needs.

Keywords: fast deployment, code generation, model based, wireless, sensor network

1. Introduction

Wireless security solutions are gaining more and more popularity because of their flexibility, convenience and clean installation, especially in home security and alarm systems. Sensor networks, however, may have an important role in complex, integrated security systems as well, where protection is provided against a wide range of threats, like physical attacks (e.g. robbery, flood), logical attacks (e.g. unauthorized access or data theft), or human/organizational threats (e.g. suspicious behavior of an employee or negligence of safety regulations). The application of wireless sensor networks is especially desirable in fast deployment scenarios.

In this paper a security architecture and a corresponding model-based code generation scheme is proposed, which provides easy and fast deployment for applications with various sensory needs.

1.1 Fast Deployment Scenarios

In certain applications the lifetime of the system is comparable with the time the setup of the system requires. Such application scenarios can be concerts, expos, conferences or exhibitions, requiring security services for a limited time only. Building the system from predefined elements and templates fastens the construction of the system. Using easily configurable elements which are easy to deploy fastens the setup of the system.

A very expensive and time consuming part of system setup is the deployment of sensors. Special security needs require special hardware; especially the wiring of the sensors can be costly and difficult. E.g. fire alarm sensors require special fireproof wiring not easily fitted into the given environment (e.g. in a protected 500-year old Town Hall drilling will not be welcome). There are more and more security solutions based on wireless sensors, where only the power supply must be provided (to charge the batteries on the sensors), or even this can be omitted if batteries with sufficiently long lifetime are used. These sensors communicate on various wireless channels, e.g. using WiFi/802.11, Bluetooth/802.15.1, or Zigbee/802.15.4 standards. The emerging Zigbee, especially tailored for communication between sensors provides an ideal architecture for mobile security applications.

Security applications require special security measures on the communication channels as well. While in the case of wired sensors usually a few wires are used and their intactness can easily be checked with proper design of the electronic components, systems with wireless sensors must be protected against new types of attack, e.g. jamming (detectable and a countermeasure alarm can be set), or message forgery (message authentication can provide strong protection).

1.2 Related work

In the field of intelligent sensors much research has been done because of the large industry demand. As a result, the IEEE 1451 Standard for Smart Transducer Interface for Sensors and Actuators is under preparation [11], [9]. Smart sensors not only measure a physical quantity (e.g. voltage or resistance) but convert the measured value to the desired unit of the physical attribute being measured. IEEE 1451 compliant smart sensors and instruments can work together with ease in a plug and play fashion. A future section of the standard (1451.5) will cover wireless communication issues as well (WiFi, Bluetooth, and Zigbee).

Smart sensors can identify themselves via the built-in Transducer Electronic Data-sheet (TEDS). The TEDS contains basic data that identifies the sensor (e.g. manufacturer ID, serial number), extended data to store all the electrical and physical proper-ties (e.g. min/max range, sensitivity), user data (e.g. sensor location) and sensor specific template data.

Model-based code generation has many appealing properties vs. handcrafted coding: the systems can be formally specified, formal verification procedures are available, and through automatic code generation coding errors can be eliminated. Based on the used initial model and the transformation tools different approaches have been used. Graph transformation is a general tool to convert the initial model into the desired format [3]. State machines [1] and hybrid models [2] proved to be rich and efficient modeling
environments for automatic code generation. Programming with design patterns also has been successfully automated using application specific models [4]. Template-based code generation is widely used due to its simplicity and feasibility [10].

2. Security System Architecture

The architecture of a sensor network based security system can be seen in Fig. 1. The wireless sensors sense their environment and send measurement data through the ad-hoc wireless network formed by the sensors (and possibly actuators) to the sensor gateway(s). The gateway provides interface between the wireless network and the Security Center. The actuators can also be operated through the wireless network: they receive their control signals through the sensor gateway and the ad hoc network. Note that intelligent wireless sensors and actuators are much more alike than their wired counterparts: sensors in addition to sending measurement values also receive information (e.g. control signals from the central system) while the actuators also send messages (e.g. status info) to the central station, besides receiving commands. In the following we handle sensors and actuators in the same way and for simplicity we refer to them as sensors.

The sensors can be configured to send measurement data in different ways: a simple solution is polling, when the central station asks the sensors to send their measurements whenever they are needed. Autonomous periodic data sending can also be provided (e.g. send data in every second). The event-based mechanism sends data only if “interesting” events occur (e.g. door opened, smoke sensed, temperature too high, etc.) In addition to measurement messages control/monitor messages are also traveling through the network: the central station can configure sensors, and they can send status information (the most important being heartbeat messages reporting that the sensors are still functional).

Most traditional sensors used in security systems measure physical quantities, e.g. thermometers, door controllers, proximity readers, fingerprint readers, smart card sensors, cameras; but much more sophisticated “sensors” can also be used in integrated security environments, e.g. picture recognition subsystems or log-based attack detection/prediction systems, just to name a few. To provide the ease of wireless deployment, the sensors (simple and sophisticated as well) are connected to the standard Smart Sensor Interface (see Fig.1), which provides various middleware services on networking and application-specific levels:

- ad-hoc network management,
- message routing/authentication,
- interfacing the sensor to the network,
- providing intelligent sensor services (e.g. periodic data sending),
- intelligent sensor configuration.

The smart sensor interfaces are inexpensive components containing a simple (8-bit) microcontroller and a radio chip (e.g. Chipcon2420 [5], Xbee [12], em250 [6]). This simple inexpensive architecture (similar to smart dust approach) enables the use of many, possibly redundant sensors providing a rich sensory field.

The wireless network is connected to the Security Center (which in fact can be centralized or distributed) through the Sensor Data Gateways. Through these gateways the Security Center unit can receive data from and send data to the network. Redundant gateways can be provided to increase fault tolerance. Naturally, different network types can be connected through different gateways.

Figure 1. Structure of the wireless-enabled security system. The Smart Sensor Interfaces provide interface to the sensors and through the ad-hoc wireless network forward messages to the Sensor Data Gateways. The gateways translate sensor data to the Security Center and also forward control signals back to the network.

Various applications require different amount and types of sensors. To facilitate fast deployment and fast system setup smart sensor interfaces are standard units: the hardware interface provides connection possibilities to various types of sensors (e.g. AD/DA, digital inputs, serial port, I2C, etc.), and most software components are also standard (e.g. network-specific components, generic sensor handling). Based on the actual requirements, sensor-specific software components will be generated and configured automatically based on sensor descriptions, which will discussed in Section 3.

The Sensor Data Gateway provides a standard interface to the Security Center. The interface is built from generic components and sensor-specific parts. The generic components are not changed; only sensor specific information is added compile-time or run-time to the gateway to enable handling of various types of sensors. The sensor specific information may be added to the gateway in two ways:

- “electronic data sheet” approach: each sensor stores its own attributes and sends this information during network setup phase to the smart sensor gateway and thus to the
Model based code generation approach for fast-deployment wireless security applications

central processing unit. With this solution all the sensor-specific information is stored in the sensors and can be accessed when the sensor is connected to the network.

– “handbook” approach: the sensor’s attributes are written in a “book” (implemented as a file). Large amount of information can be stored in the handbook; no network communication is required and the limited resources on the sensors are not wasted. The book can be accessed by various components, e.g. smart sensor interfaces can be generated based on the sensor description, or the sensor data gateway can be configured using relevant data stored in the handbook.

Although both solutions are technically possible, in the proposed security system architecture there is no specific need for the more sophisticated “electronic data sheet” approach: various system components can easily access sensor-specific information from the handbook and this approach suits better the limited resources of the sensor interfaces. In the following examples the handbook approach will be used.

3. Sensor Description

The description of the sensor contains all the necessary information to handle the sensor’s hardware as well as rules to convert and use the sensor’s measurement values. The functional description contains high level information for the gateway and the central processing unit, while the smart sensor interface is generated from the platform specific hardware and software description.

To ensure uniform handling and portability of the data, XML file format is used, shown in an XML editor in Fig. 2. In the example a simple temperature sensor is defined for a MicaZ mote [8] using TinyOS operating system and nesC [7]. The description contains high-level (functional) and low-level (platform specific) information as well.

![Figure 2. The description of a temperature sensor in xml format](image)

3.1 Functional Description

The final consumer of the measurement data is the central processing unit. The sensor data gateway converts raw data to standard formats understood by the central unit, based on the functional description of the sensor. These data fields identify the sensor type, the data format, accuracy, etc. This field also contains conversion information (how to interpret the bit series of the sensor) and display options for graphical representation as well. The tags identify the sensor type (Temperature) with code “7” to identify this type of measurements, measurement units (Celsius, Kelvin, and Fahrenheit), the accuracy of the sensor, and the conversion formulae to generate the required units from the raw data. The optional image fields can define various graphical representations for the sensor.

3.2 Platform Specific Description

The platform specific sensor description defines low level interfacing to the sensor. It includes hardware specific information (e.g. at which port the sensor can be accessed), usage rules (e.g. how the measurement values can be read), architectural information (e.g. module descriptions, wiring), and any other relevant platform-specific information (e.g. constants, functions, any necessary glue code). Fig. 3 illustrates two fields in the temperature sensor’s description for the Mica mote platform using TinyOS operating system and nesC language: wiring and interface definition.

The wiring information inserts the module to the smart sensor interface, using the defined interfaces. The init, read, and stop fields describe how to start, operate, and stop the sensor, respectively (shown in Fig. 2). Naturally these fields must be platform specific and have to contain information for the code generator.

![Figure 3. Interface description of the temperature sensor on Mica mote HW and TinyOS/nesC environment](image)

4. Code generation

The code generation is based on the sensor descriptions and code templates, as shown in Fig. 4. The Smart Sensor Interfaces and the Sensor Data Gateway are implemented on their specific platforms, using meta tags in the code to identify sensor specific code segments to be generated. The translator, using the Sensor Description generates and inserts the appropriate code into the templates.

The translator is platform independent and can understand generic commands like replace code segment and make new instance of a code segment. Using these meta-commands (recursively, if necessary) the translator can generate the required Smart Sensor Interface and Sensor Gateway Code. The codes then can be compiled and downloaded to the desired target using standard make utilities and the system is ready to run.
The following example shows a Smart Sensor Interface template code segment with meta tags. The `<MakeTimer>` and `</MakeTimer>` tags instruct the translator to generate a code segment for all sensors, while the `<$...$>` tags are replacement instructions:

```c
<MakeTimers>
  event result_t <$SType$>Timer.fired(){
    <$FnCall$>
    return SUCCESS;
  }
</MakeTimers>
```

The code is generated for two embedded sensors, a light and a temperature sensor. The generated code segment includes two instances of the event handler, one for each sensor, with the correct function calls and references:

```c
event result_t TemperatureTimer.fired(){
  call TempControl.start();
  call Temperature.getData();
  return SUCCESS;
}
event result_t LightTimer.fired(){
  call PhotoControl.start();
  call Light.getData();
  return SUCCESS;
}
```

5. Results

We implemented a wireless sensor network based security surveillance system as a case study. The system contains MicaZ motes as sensors [8] running TinyOS/nesC [7], and a PC serves as gateway. The motes can control several sensors, e.g. temperature and light sensors, accelerometers, magnetometers, etc. The Smart Sensor Interface code running on the motes was generated automatically for each unit, implementing various sensors. The gateway in the example does not require compile-time code generation; it uses the functional sensor description run-time (e.g. for unit conversion).

The Smart Sensor Interface code is illustrated in Fig. 5 showing its component structure, automatically generated by the TinyOS documentation tool. The initial template code is shown in Fig. 5(a). The figure shows the standard components (e.g. Main) and their connections to each other through interfaces (e.g. StdControl). In Fig 5(b) the structure of the generated code is shown when a light sensor and a two-axis accelerometer are used. New components handling the sensors were automatically added to the code (PhotoTemp, Accel, and three new instances of TimerC), and were connected to standard components via appropriate interfaces. Note that Fig. 5 shows the structural modification of the code, while the change of the internal content of component GeneralM was illustrated by the examples in Section 4.
Model based code generation approach for fast-deployment wireless security applications

containing a photo sensor and a two-axis accelerometer.

Internal changes in the components are not shown

The GUI of the surveillance application is shown in Fig. 6. The Tree Manager window displays the topology of the ad-hoc network, along with the pictograms of the available sensors; e.g. Mote 5 contains a microphone, a light sensor, and a thermometer. The GUI allows the configuration, control, and query of individual sensors. The system can automatically identify new sensors in the network and can detect sensor failures as well. The gateway also generates and forwards events (in the example to the surveillance GUI) when a sensor appears in or disappears from the network. The active motes and their installed sensors can be accessed and controlled through the gateway, as shown in the control window of Mote 5.

![Figure 6. Elements from the GUI of the surveillance application. Smart sensor interfaces providing ad-hoc wireless access to sensors were generated from platform-specific sensor description. The gateway uses functional-level sensor information to display temperature values (see the defined units and conversion formulae in Fig. 2)](image)

6. Summary

In integrated security systems wireless solutions will be increasingly important because of their flexible, fast and convenient installation. Such systems can provide protection against various threats and attacks, with safety requirements similarly high to that of the wired solutions. In fast deployment systems, where fast and cost effective deployment is crucial, the application of wireless sensors can radically shorten the deployment time of the system.

In this paper a wireless-enabled security architecture and a corresponding model-based code generation scheme was proposed. The architecture handles various types of sensors in a similar manner through the generic Smart Sensor Interface. The interface provides the necessary middleware services to enable the access of sensors via the ad-hoc network, including routing, state monitoring, message authentication, and naturally, sensor interfacing. Wireless Gateways connect the Security Center (not discussed in this paper) to the ad-hoc network with generic sensor interfaces.

The fast as error-free development and deployment of security applications is eased by the proposed automatic code generation of various sensor-related components. The Smart Sensor Interfaces and the Wireless Gateways can automatically be generated from sensor description information. The sensor descriptions can also be used runtime by higher level components.

The architecture and the code generation scheme were illustrated in a case study, where an ad-hoc sensor network provided measurements. The code for the Smart Sensor Interfaces was generated automatically based on platform-specific sensor description. In the test application the usage of functional sensor description information was also illustrated.

Acknowledgment

This research was partially supported by the Hungarian Government under contract NKFP2-00018/2005.

References

Author Biographies

László Szabados got his BSc in Information Technology from the University of Pannonia, Veszprém, Hungary, in 2008. Currently he is an MSc student at the same university.

András G. Tóth got his BSc in Information Technology from the University of Pannonia, Veszprém, Hungary, in 2008. Currently he is an MSc student at the same university.

Gyula Simon got his MSc and PhD in electrical engineering from the Technical University of Budapest, Hungary, in 1991 and 1998, respectively. Currently he is an associate professor at the University of Pannonia, Veszprém, Hungary.