SMEPP: A Secure Middleware For Embedded P2P

Rafael J. CARO BENITO[1], Daniel GARRIDO MÁRQUEZ[2], Pierre PLAZA TRON[3], Rodrigo ROMÁN CASTRO[2], Nuria SANZ MARTÍN[3], José Luis SERRANO MARTÍN[1]
[1]TECNATOM, S.A., Av.Montes de Oca, 1, S. Sebastián de los Reyes (Madrid), 28703, Spain
Tel: +34 916598600, Fax: + 34 916598677, Email: rcaro, jlserrano@tecnatom.es
[2]University of Málaga, Campus de Teatinos S/N, Málaga, Spain
Tel: +34 952133371, Fax: + 34 952131397, Email: roman, dgarrido@lcc.uma.es
[3]Telefónica I+D, C/ Emilio Vargas, 6 Madrid, 28043, Spain
Tel: +34 913374000, Fax: + 34 913374004, Email: pierre,nsanz@tid.es

Abstract: The increasing presence of embedded devices with internet access capabilities constitutes a new challenge in software development. These devices are now cooperating in a distributed manner towards what has been called as "Internet of Things". In this new scenario the client-server model is sometimes not adequate and dynamic ad-hoc networks are more common than before. However, security poses as a hard issue as these systems are extremely vulnerable. In this paper, we introduce SMEPP project, which aims at developing a middleware designed for P2P systems with a special focus on embedded devices and security. SMEPP is designed to be deployed in a wide range of devices. It tries to ease the development of applications hiding platforms details and other aspects such as scalability, adaptability and interoperability. A full implementation of this middleware is already available that incorporates security features specially designed for low-resource devices. Moreover, we describe two business applications being developed using this middleware in the context of "Digital Home" and "Environmental Monitoring in Industrial Environments".

Keywords: Middleware, security, P2P, embedded.

1. Introduction

Embedded peer-to-peer (EP2P) systems are emerging as a new scenario where limited resource devices are accessing the network. One of the barriers preventing wider and faster adoption of these systems is their intrinsic complexity. A middleware providing abstraction is needed to ease application and service development. This middleware should include mechanisms for secure interaction between peers and abstract developers from problems such as lack of infrastructure or security vulnerabilities. It should also allow interaction with third parties. The expected results of this middleware would therefore be a cost reduction in the development of applications and services for EP2P systems.

European Commission's FP6-funded STREP project SMEPP (Secure Middleware for Embedded Peer-to-Peer systems) [1] tries to develop such a middleware. It is specifically designed for deployment in a range of different devices, from wireless sensor networks [2] to full-featured computers. Energy consumption is therefore a key issue from the beginning. Security is also considered from the design phase, including cryptography and prevention of attacks, specifically those that are most relevant for resource-constrained devices. Additionally, it interoperates with existing third party applications. This middleware is being validated through its use in two business applications, leaving the client/server paradigm to take advantage from the resources in the network.

1.1 – Related work

Related work can be found in some projects. RUNES[3] project developed a software architecture for embedded systems with network capabilities. It also introduces a component model for the development of the architecture including an API that hides the details of hardware that are specific to the device. MORE [4] is another research project that proposes a service-based architecture similar to SMEPP. This architecture considers three levels of devices including sensor networks, mobile devices and desktop computers. MORE also includes simplified versions of some protocols for less-capable devices. Finally, AMIGO [5] is oriented towards smart computing for home systems, smoothing the integration of services in electronic devices using discovery mechanisms.

All these projects have been considered. However, none of them covers all the topics relevant to SMEPP, such as security, peer-to-peer communication, services, etc.

2. Objectives

The main objective of the SMEPP project is to develop a new middleware, based on a new network centric abstract model, specially designed for embedded devices using the peer-to-peer paradigm, and trying to overcome the main problems of the currently existing domain specific middleware proposals. The middleware will be secure, generic and highly customizable, allowing for its adaptation to different devices and domains.

In order to achieve these generic objectives, the project research efforts and specific project goals have been arranged around four main topics:

- Abstract Service and Interaction Model.
- Middleware Architecture and Infrastructure.
- Security.
- Applications.

3. Innovation

The study of the state of the art of middleware for EP2P systems [6] allowed detecting necessities in this field which had not been covered by previous works. In addition, SMEPP tried to use the best of the different existing approaches obtaining a solution for the development of software in EP2P systems. Special attention has been paid to the WSN and MANET technologies. Here is a summary of the strongest points of SMEPP:

- Security inside the middleware: Security has been considered at all levels of the middleware, from the interaction model to the lower levels.
- Interoperability with internet standards/legacy systems.
- Adaptation and configuration to different devices/OS/ platforms: networking interfaces and protocols, operating systems, hardware platforms and programming languages.
- Real-Time Requirements: maximum transmission rate, maximum amount of time that the channel is being used during each transmission, etc.
- Other QoS Requirements: SMEPP can monitor the quality of resources and adapt, depending on application constraints.
- Energy Awareness: The SMEPP architecture provides schemes to help developers define energy requirements and schedules.
- Scalability from design.
- Requirements on the hardware: SMEPP can be executed in small devices such as sensor nodes or in higher computer devices.

4. Technology Description

The work has been split into several work-packages. The most relevant activities for the purpose of this paper are architecture, security and validation applications.

4.1 – Architecture

The software architecture for SMEPP middleware follows a component model. It also contains a set of tools for adapting the middleware to different devices, applications and networks. The idea is to produce a component framework where each component can be efficiently adapted to its environment.

Two key aspects have been considered: the management of security aspects and keeping the architecture flexible, scalable and adaptable to different devices and platforms. Architectural drivers were defined: Security, adaptability, scalability, interoperability and platform heterogeneity.

Interaction with third party systems, such as OSGi [7], has been taken into account. As a result, SMEPP and OSGi coexist and cooperate in the same application.

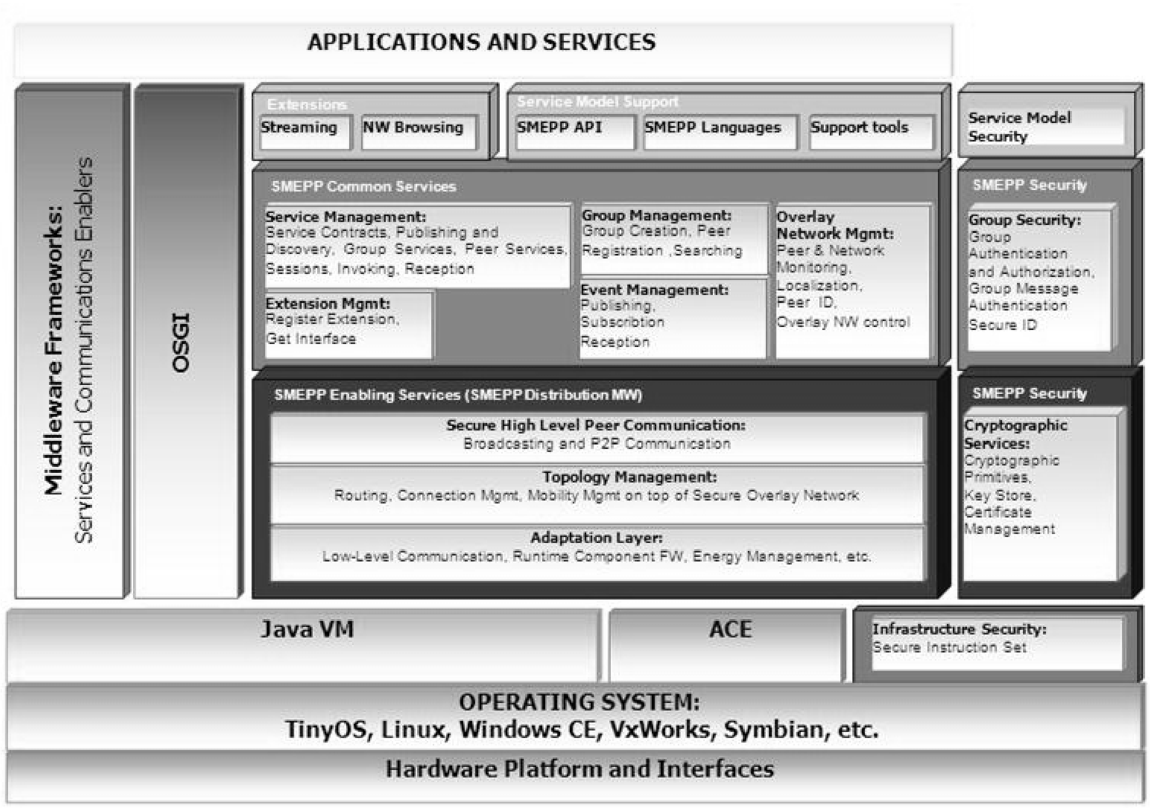
APPLICATIONS AND SERVICES
Middleware Frameworks: Services and Communications Enablers
OSGI
Extensions
Streaming
NW Browsing
Service Model Support
SMEPP API
SMEPP Languages
Support tools
Service Model Security
SMEPP Common Services
Service Management: Service Contracts, Publishing and Discovery, Group Services, Peer Services, Sessions, Invoking, Reception
Group Management: Group Creation, Peer Registration ,Searching
Overlay Network Mgmt: Peer & Network Monitoring, Localization, Peer ID, Overlay NW control
Extension Mgmt: Register Extension, Get Interface
Event Management: Publishing, Subscribtion Reception
SMEPP Security
Group Security: Group Authentication and Authorization, Group Message Authentication Secure ID
SMEPP Enabling Services (SMEPP Distribution MW)
Secure High Level Peer Communication: Broadcasting and P2P Communication
Topology Management: Routing, Connection Mgmt, Mobility Mgmt on top of Secure Overlay Network
Adaptation Layer: Low-Level Communication, Runtime Component FW, Energy Management, etc.
SMEPP Security
Cryptographic Services: Cryptographic Primitives, Key Store, Certificate Management
Java VM
ACE
Infrastructure Security: Secure Instruction Set
OPERATING SYSTEM: TinyOS, Linux, Windows CE, VxWorks, Symbian, etc.
Hardware Platform and Interfaces

Figure 1: SMEPP Middleware Architecture in context

Figure 1 shows a high-level view of the architecture. It is comprised of three functional layers. Below these three layers there is an execution environment allowing all architecture components to run. In the upper layer, SMEPP defines an abstract service model, on top of which the applications are built. The main role of the Service Model Support block is to provide the application developer with an API to access the middleware. The second block in this upper layer is responsible for supporting extensions. These extensions allow the service model to be enhanced with new functionalities and domain-specific features.

The intermediate layer contains the SMEPP Common Services. It supports the core functionality: event, group, service, message management, or network monitoring. This layer uses a component-based technology. Different implementations of the same component can exist for different devices or environments. Components in this layer are:

- Event Management: This component allows for creating, publishing, receiving or subscribing events to the SMEPP network.
- Group Management: This component is responsible for handling groups inside the middleware. The group concept is a key concept in SMEPP, as it glues peers together in a basic entity. Peers are authenticated and authorised when entering a group.
- Service and Message Management: This component manages service contracts. It also supplies the mechanisms for message exchange among peers and/or services.
- Extension Management: It provides the tools to add extensions to SMEPP (e.g. OSGi).
- Overlay Network Management: It is responsible for the acceptance of new peers to the network. It also monitors of peer status, and notifies of peer (dis)connection.

At a lower level we find the SMEPP Enabling Services. They include basic peer-to-peer communication and underlying communication protocols, (e.g. secure routing). These components depend heavily on the restrictions imposed by the infrastructure. The three components available at this lower level are:

- Secure High-Level Peer Communication: It offers the above layer the ability to communicate peers in a secure and abstract way.
- Secure Topology Management: It is in charge of security in the SMEPP network, managing authentication of new peers, permissions to access the SMEPP network and protection of routing information exchanged in the network.
- Adaptation layer: It is responsible for providing an abstract interface above the execution infrastructure. It isolates the above layers from implementation details. It offers primitives for sending messages and energy management.

Security is taken into account from the start in the software architecture. It covers all the architectural layers. Built-in security components in SMEPP architecture are:

- Group Security: Maintains security inside SMEPP groups. It is responsible for tasks such as peer authentication trying to join a group.
- Cryptographic Services: Provides cryptographic primitives (encryption, decryption, digital signature, etc.)
- Infrastructure Security: It uses special built-in features from specific devices. For instance, it can take advantage of cryptographic primitives available in some processors.

4.2 – Security

Security is a critical issue for the correct performance of a embedded P2P network. This is especially important in low-end devices as they are particularly weak when confronted to external or internal security threats. One of the main goals of SMEPP is the seamless integration of security mechanisms inside middleware architecture. These security mechanisms must be transparent and adaptable.

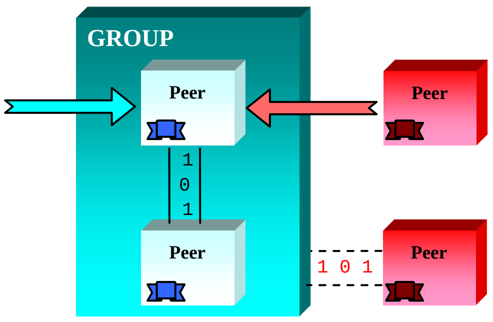

Figure 2: Group-oriented security

Transparency is achieved by integrating security as an intrinsic part of group functionality inside SMEPP (see figure 2). When joining a group, a peer must present some security credentials. If they are valid, the peer will be allowed to access the secure communications taking place inside the group. The application developer must only provide those credentials when trying to join a SMEPP group. The middleware will be in charge of implementing this admission mechanism and the secure communication inside the group.

When a peer tries to join a group, the middleware executes a mutual authentication mechanism based on challenge-answer protocols. A shared session key will provide security for the communication channel. This session key is shared by all members of a group and used by the cryptography symmetric-key primitives and hash functions. At the same time, other transparent protection mechanisms allow to renew the session key when necessary, detecting malicious behaviour of a group member, etc.

	Level 0	**Level 1**	**Level 2**
Group admission	None	Shared secret key	Public key cryptography
Data security	None	Authentication	Authentication and encryption
Session key protection	None	Global (key renewal)	Global and Local (anti-SCA)

Table 1. Security levels inside SMEPP

Respecting adaptability, security is not imposed to the developer. The desired security level can be established (see Table 1). SMEPP allows configuration of the group admission process and the security in information transmission as well as session key protection, including protection against side channel attacks (SCA).

SMEPP offers an additional tool for protection of communications: security domains. A SMEPP network contains multiple security domains: A global domain (members of a SMEPP network) and several group domains (members of a SMEPP group). An application can make a SMEPP network accessible by any device or only some privileged devices.

Resource-constrained devices, such as smart sensors are able to execute symmetric cryptography via hardware [8] or software [9], and can also execute public key cryptography through the use of elliptic curves [10]. Finally, as SMEPP follows a component-based model, it is possible to dismiss security features not required in the final deployment.

5. Business impact

Reduction of the life cycle time of software is a trend in the Internet and embedded domain. This means that ROI must be obtained very quickly. A middleware abstracting the wide range of OS and devices is required to speed up the process of development. This reduction in development time can be translated directly to cost reduction. The training cost of the developers will also decrease. Finally, when the need to tackle security and privacy issues is mandatory, SMEPP appears as a very appealing solution [11, 12].

SMEPP middleware looks promising for software development enterprises, from companies specialized in small devices (e.g. sensors) to network operators. In the second case, the interest comes from the development of applications for mobile devices with open source operating systems (e.g. LiMo [13], Android [14], OpenMoko [15]).

6. Applications

Two application domains have been considered for validation purposes: Environmental control in industrial environments and services for digital home using mobile devices. Similar high-level requirements are found in both fields such as device management, alarm generation, video-conference, message exchange, etc. Events can be generated in both domains either by devices or by users.

Both validation scenarios also have similar security and privacy requirements. They need the notion of ‘groups’, where users need to authenticate themselves in order to access to their services. In addition, all the information exchanged within the group must be adequately protected. Finally, when sensitive information is processed (e.g. radiation dose, patient data), a higher security level can be requested to the middleware.

6.1 –Environmental and radiological monitoring in industrial plants

This application [16] consists of two parts, the first one for Remote Control of Work and the second one aimed at the Radiological Environmental Monitoring.

The first part integrates data from area and personal dosimetry sensors in a single tool, displaying all the available field information, including video and audio, as well as other features (access to documentation, personal identification by biometric systems, etc.). This means a breakthrough in nuclear safety, improving the radiological protection of workers. It also is an efficiency improvement, with better control and supervision of work. The system applies directly to various business areas in the nuclear sector.

The second part (Radiological Environment Monitoring) allows the integration of radiation measurements along with other variables of interest (primarily meteorological). This part of the application also has a direct translation in other fields such as environmental protection, infrastructure, facing the threat of terrorism or the smuggling of radioactive materials.

A first version [17] with three levels of devices for processing information has been developed: (1) a network of Crossbow micaZ motes using 802.15.4, (2) PDA with WiFi (802.11b / g) for itinerant workers and (3) laptop with WiFi for Radiological Protection personnel responsible for monitoring environmental conditions and radiological equipment.

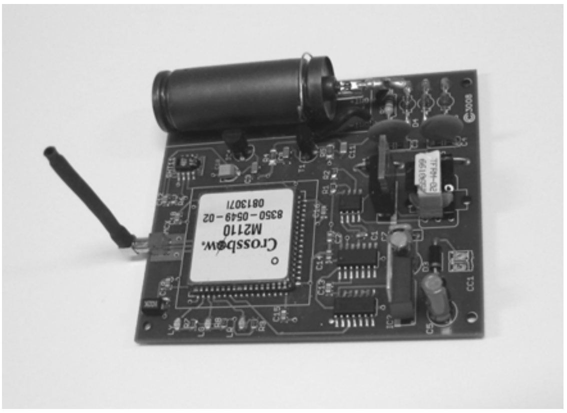

Figure 3. Prototype module integrating simple radiation sensor

The current version includes a radiation sensor with a wireless transmitter (see Figure 3). At the application level, transmission of audio and video is also being included as well as work-flow features. The software is deployed in a wireless sensor network supporting groups of workers and staff of Radiological Protection. The goal is to reduce the radiation received by staff and enable collaboration between teams working in harsh environments.

6.2 –Services for digital home systems

In the domain of digital home services, several applications are being developed. One of them allows users to generate some events, such as alarms. The system can locate other users that are best suited to manage the alarm depending on the context and route the alarm to that user. If the alarm can not be handled by that user, the system automatically tries to find a new user that can respond to the event. Once the alarm is received, users can exchange text messages, send images or establish a video-conference.

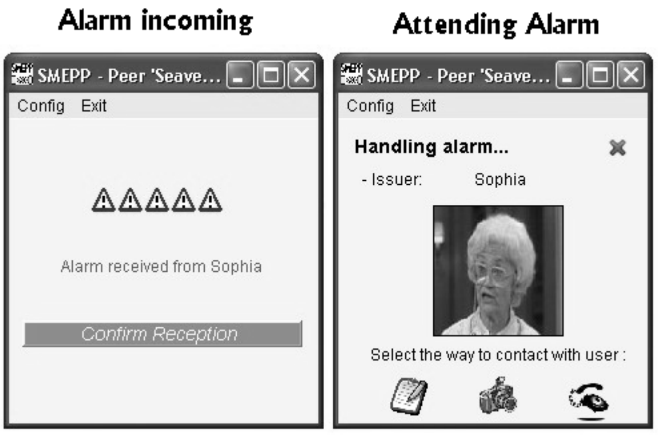

Figure 4. Application prototype with alarm management and communication between relatives

In the current version, users can generate alarms through a simple graphical interface in a mobile device such as a PDA (see figure 4). This alarm is received by the rest of users in the network. When a user confirms reception of the alarm, both users establish communication. Other users are notified that the alarm is being handled by someone else.

In the version of the application being developed at the moment, the alarm will not reach all the users. The system will detect in real time in a context-sensitive way who is the optimal user that can manage the alarm. In this way, the performance improves and users are not informed of alarms that are of no relevant.

Alarms can be generated also by sensors that communicate with the residential gateway. This gateway forwards the information to SMEPP network. Sensors range from water or gas leak detectors to biomedical devices measuring Blood pressure.

7. Conclusions

The amount of research works on middleware aiming at efficiency and reuse in software development in the last years is very important. However, it is not usual to focus on devices with few resources. At the same time, the number of “simple” devices with network access is increasing at an enormous speed. If we try to find middleware focused on limited-resource devices and peer-to-peer communication in an ad-hoc network, the number of results with a functional implementation is very small.

Security is one of the challenges in the near future for this kind of systems. A network that is created dynamically without an underlying infrastructure or servers can not rely on standard security schemes.

In this paper, we have shown a high level view of the SMEPP project, where a middleware for secure peer-to-peer communication among embedded devices has been designed and built for development of secure embedded P2P applications. Currently, a full implementation of the middleware is available, and work is still in progress with performance improvements. The chosen language for the implementation is Java. Interoperability with .Net is also provided. There is also a version in nesC[18] for TinyOS[19], called "SMEPP Light". SMEPP is running in devices such as smart sensors, smart phones, PDAs, laptops or desktop PCs. The two validation applications are currently being developed in an iterative approach. Prototypes of both applications are available.

From the industrial point of view, the most interesting advantage of SMEPP middleware is the possibility of abstraction for embedded P2P applications. This is not the case with proprietary systems where each implementation requires an ad-hoc development.

References

[1] SMEPP Project website. http://www.smepp.org

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "Wireless sensor networks: a survey". Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 38, no. 4, pp. 393-422, March 2002.

[3] Costa, P. et al. The RUNES Middleware: A Reconfigurable Component-based Approach to Networked Embedded Systems. 16th Annual IEEE Internacional Symposium on Personal Indoor and Mobile Radio Communications IMRC'05), Berlin, Germany. September 2005.

[4] MORE project, http://www.ist-more.org/

[5] Georgantas, N.; Ben Mokhtar, S.; Bromberg, Y.; Issarny, Valérie; Kalaoja, J.; Kantarovitch, J.; Gérodolle, A.; Mevissen, R. 2005. The amigo service architecture for the open networked home environment. Proceedings of 5th Working IEEE/IFIP Conference on Software Architecture. WICSA. Pittsburgh, 6 - 10 Nov. 2005

[6] SMEPP Deliverable D1.1 State of the art and Generic Middleware Requirements at http://www.smepp.org

[7] OSGi website. http://www.osgi.org

[8] IEEE 802.15 WPANTM Task Group 4 (TG4). http://www.ieee802.org/15/pub/TG4.html

[9] K. Jun Choi, and J.-I. Song. "Investigation of Feasible Cryptographic Algorithms for Wireless Sensor Network". Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT 2006). Phoenix Park (Korea), February 2006.

[10] An Liu, and Peng Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks". Technical Report TR-2007-36, North Carolina State University, Department of Computer Science, November 2007.

[11] ITU Internet Reports 2005: The Internet of Things. November 2005

[12] Internet of Services and Internet of Things: adapting to user, task, and location in a seamless fashion. Fia Madrid. December 2008

[13] http://www.limofoundation.org/

[14] http://code.google.com/intl/en/android/

[15] http://wiki.openmoko.org

[16] E. Cabrera, R.J. Caro, M. Díaz, J. Serrano "Proyecto SMEPP: Redes inalámbricas de sensores y sistemas "peer-to-peer" empotrados. Aplicación en la industria nuclear" Proceedings of the 33rd meeting of the Spanish Nuclear Society (SNE 2007)

[17] R.J. Caro "SMEPP un proyecto I+D+i del 6º PM aplicado al Control Radiológico Medioambiental y a la mejora de la Protección Radiológica" Journal of the Spanish Radioprotection Society. Nº 54, vol. XIV, 2007.

[18] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, D. Culler, "The nesC language: a holistic approach to networked embedded systems", in: Proceedings of the ACM SIGPLAN conference on programming language design and implementation (PLDI 2003), San Diego, CA, USA, June 2003, pp. 1–11.

[19] TinyOs Community Forum, http://www.tinyos.net