MAIPAN – Middleware for Application Interconnection in Personal Area Networks

Miklós Aurél Rónai, Kristóf Fodor, Gergely Biczók, Zoltán Turányi and András Valkó
Ericsson Research, Traffic Lab,
P.O.Box 3, 1300 Budapest, Hungary
E-mail: {Miklos.Ronai, Kristof.Fodor, Gergely.Biczok, Zoltan.Turanyi, Andras.Valko}@ericsson.com

***Abstract*— This paper proposes a middleware that creates a dynamically changeable, but uniform computing environment for personal area networks (PANs). The middleware hides the physical scatteredness and the dynamically changing configuration of the PAN and presents its capabilities as a single computer. Using the uniform application programming interface (API) offered by the middleware, developers creating distributed applications do not have to take care of various PAN configurations or PAN dynamics. The solution provides easy set-up of PANs and transparent redirection of ongoing data flows. The proposed middleware interconnects services offered by applications running on different devices by creating virtual channels between the input and output outlets of the applications. The middleware is able to reconfigure these channels when the configuration or user needs change.**

I. Introduction

The ever-growing number of wireless terminals, such as smart phones, personal digital assistants (PDAs) and laptops, raises the need to set up, configure and reconfigure personal area networks (PANs) in an easy and ergonomic way.

This paper addresses the goal of creating a dynamically changeable, but uniform computing environment for PANs, which integrates wireless and wired, stationary and mobile devices that are connected to each other. The paper presents the Middleware for Application Interconnection in Personal Area Networks (MAIPAN) solution—which is situated below the applications and above the network layer—that hides the individual devices participating in the PAN and presents the capabilities of applications running on the devices as if they were located on a single computer. This provides a standard ”PAN programming platform”, which allows the easy set-up of personal area networks and dynamic connection and disconnection of applications running in the PAN. Application programmers using the uniform application programming interface (API) offered by the middleware can develop software without taking care of the various PAN configurations or PAN dynamics. They can assume certain capabilities, but disregard whether these capabilities are provided by one application running on one device or by a set of applications running on several devices. They only have to register the inputs and outputs of their applications in the middleware, and they do not have to take care of which kind of devices or applications will be connected to these outlets and will use their programs.

The presented middleware does not include solutions for service discovery, physical, link or networking layers, but assumes that such are already available. The middleware does not contain intelligent functions, which can act on behalf of the user, but this kind of solutions can be added later, if necessary.

An example for a PAN application can be seen in Fig. 1. Assume a MAIPAN-aware PDA user, who enters a MAIPAN-aware room where speakers are placed in the corners. The user decides to listen to music, but she neither has mp3 files nor an mp3-player. She turns on her PDA to check what kind of devices and services are available in the room. First, independently from MAIPAN the PDA's networking protocol establishes network connections to the devices located in the room, and the service discovery protocol (SDP) looks for services offered in the environment. After these steps she can configure with the MAIPAN dispatcher a PAN, which contains her PDA, the speakers in the corner, a fileserver and an mp3 player that are available through the network. MAIPAN establishes the necessary connections, that is, it connects the fileserver to the mp3 player's file input, the sound output of the mp3 player to the speakers and the control input of the mp3 player to the PDA. From now on she can select the mp3 file on the fileserver she wants to listen to, and instruct the mp3 player via her PDA to play the music on the speakers. In the figure the arrows show the data flow between the devices.

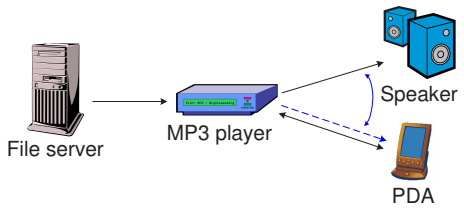

Fig. 1. Listening to music

In case she has to move from one room to another, with the MAIPAN dispatcher she can reconfigure her PAN and redirect the output of the mp3 player to the built-in speaker of her PDA. After she entered another room, the networking protocol establishes network connections and the service discovery protocol looks for available services. In case there are speakers in the other room too, she can reconfigure her PAN with the dispatcher again to continue listening to music on the speakers in the room. In the figure the dashed arrow shows the configuration after the user switched from the speakers in

the room to the speaker in her PDA.

The installation of software purchased in a MAIPAN-aware box becomes possible by adding the box to the user's PAN. Assume that the user wants to play a multi-player game with her friend in the room they are sitting. After the networking layer established the network connections, with the help of the service discovery protocol the user finds the Tetris game running on the MAIPAN-aware box. To "install" the software the user can instruct the MAIPAN dispatcher to connect one input of the Tetris-box to a PC keyboard, the other one to the keypad of her mobile phone, and the output of the Tetris-box to the large display located on the room's wall (Fig. 2). In case the friends want to leave the room, the display output can be seamlessly redirected to the PDA's internal display—by instructing the dispatcher—, and they can continue gaming. Again, in the figure the arrows show the data flow between the devices, and the dashed arrow shows the configuration after the user switched from the monitor on the wall to the internal display of the PDA.

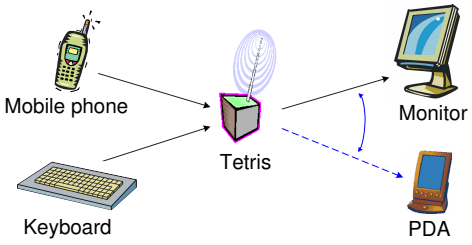

Fig. 2. Playing Tetris

As the examples show, with MAIPAN it is easier for the user to create or reconfigure personal area networks. Moreover, it is easier for the application developers to create software for PANs. Those programmers, who want to write PAN applications, only have to register the inputs and outputs of their applications in the middleware and define the types of these outlets. However, MAIPAN provides support also for those programmers who want to create PAN configurator software, i.e., dispatcher applications.

Without using the middleware, application programmers would have to handle, for example, connection and disconnection of devices, implement access control, security and transport mechanisms (in case they are not supported by the networking and transport layers below) and have to communicate with the service discovery protocol. The middleware takes away the responsibilities of handling these functions from the application developers, thus they do not have to deal with these issues.

II. Related work

Mark Weiser presented the concept of ubiquitous computing in the early 90s [1]. Ten years after Weiser's publication, Satyanarayanan discussed the vision and challenges of ubiquitous computing in [2], where he introduced the expression *pervasive computing*. Mascolo et al. in [3] discussed how a mobile computing middleware should be designed. [4] proposes the concept of Personal Security Domains (PSDs)—where a PSD consists of a user's personal devices, and investigates security issues regarding PANs. Nowadays several projects are running in these research topics.

III. MAIPAN – Middleware for Application Interconnection in Personal Area Networks

A. Basic concepts and definitions

MAIPAN distinguishes among *device*s, *application*s and *service*s. The word "device" refers to the physical device and by "application" we refer to the software that offer the "services". For example, using these abstractions in case of a mouse we can say, that the mouse is a "device" where a "mouse application" is running, which offers a "mouse service". Or another example is the mp3 player: there is a device where the mp3 player application is running, which offers the mp3 player service. The distinction between application and service is necessary, because users want to use services, they do not want to care about the application offering the service.

MAIPAN is based on three concepts (Fig. 3): *pin*s, *channel*s and *session*s. Applications offering the services have input and output outlets, which are called pins—borrowing the expression from the integrated circuit world. Pins are the connection points of the applications to the middleware, so the middleware sees the applications in the PAN as a set of input and output pins. A pin has a pre-defined type, which shows the type of data that the pin can emit or absorb, that is, the type of information the application can handle (e.g., mouse movements, keystrokes). According to the needs new types can be defined any time. The dispatcher application is responsible to connect the pins with appropriate types.

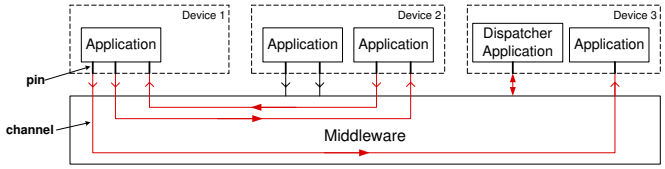

Fig. 3. MAIPAN session

To enable communication between pins, the middleware creates and reconfigures channels, which are point-to-point links that interconnect pins. The set of channels that are necessary to use a PAN service is called session. For example, the channels between the entities that can be seen in the mp3 player scenario (Fig. 1) compose the mp3 player session: there is a channel between the fileserver and the mp3 player, another channel between the mp3 player and the speakers, and a control channel between the mp3 player and the PDA. The PDA plays only the role of the *control entity*, it is the owner of the session, but it does not participate in it. Applications are neither aware of channels nor sessions, they only know about their pins.

B. Security and access control

Security and access control functions are defined and handled on device level. This means that access to services

(i.e., access to application pins) are granted for devices, thus if a device gets the right to use a given service, then all applications running on this device will be able to access the service. If we assume that in the PAN there are small devices that offer one or two simple services (e.g., mouse, mp3 player), then in this case it is simpler to grant the access of services to devices instead to each application.

The dispatcher application running on a device, which plays the role of the control entity, can set up and reconfigure sessions. The control entity has to check and ask for the necessary access rights to enable the usage of a given service for the user. For instance, this main control entity can be a PDA, which has enough computing power to manage a PAN. All other devices participating in the PAN are called *participants*. In special cases, participants may delegate the access control rights to other devices, which will be referred to as *manager*s.

Security solutions, for example, described in [4] can be used with MAIPAN to ensure secure communication between devices.

C. Architecture

Based on the concepts above we designed MAIPAN, whose architecture is depicted in Fig. 4. Applications run over the middleware, while layers under the middleware provide end-to-end routing and data transmission functions. As it can be seen in the figure, the protocol stack is vertically divided into two parts. The aim of the *data plane* is to provide effective and secure data transport between applications. The *control plane* is responsible for managing pins, channels, sessions and for handling security and access control.

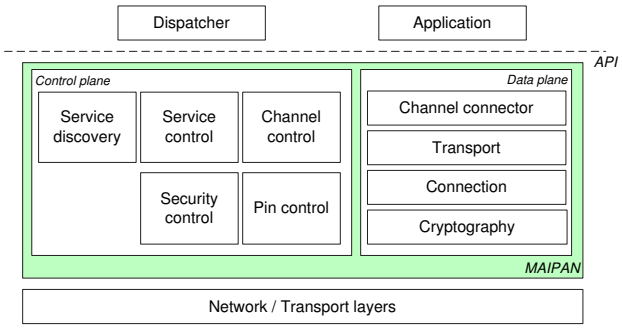

Fig. 4. Architecture of the middleware

1) Data plane: The application sends data through a pin to the middleware, where the *channel connector layer* redirects the data to the corresponding channel. The *transport layer* creates packets and provides functions such as flow control, reordering, automatic re-transmission, quality of service, etc., if these functions are not supported in the layers below the middleware, but are necessary for the application. The *connection layer* adds information to the packet, which is needed for the delivery: address of the source and destination device and the identifier of the channel. Finally, the *cryptography layer* calculates a message integrity check (MIC) value and encrypts the packet if necessary.

2) Control plane: The control plane contains the control functions which are necessary to manage the PAN. The *service control* part registers local services offered by the applications, handles their access rights and communicates with the service discovery protocol. In theory, any kind of SDP can be attached to MAIPAN, such as SLP, UPnP or Salutation [5]. The *channel control* creates and reconfigures sessions initiated by the dispatcher application. According to the needs of the dispatcher it asks the *pin control* parts of the participating devices to build up the channel between the pins. The pin control part instructs the channel connector layer to create a channel, activates the necessary transport functions for the given channel in the transport layer and sets the destination of the given channel in the connection layer. The *security control* part initiates and co-ordinates the authentication procedure between devices and stores the necessary information for communication (e.g., security keys).

IV. Conclusion and future work

This paper proposes MAIPAN, a middleware for application interconnection in personal area networks. The essence of the middleware is to create a PAN programming platform, whereby hardware and software resources are interconnected, and the scatteredness of the PAN is hidden from the services offered by applications. Every application that runs on one of the MAIPAN-enabled devices sees the other applications and itself as if they were all running on the same device. Using the proposed architecture, developers creating distributed applications for PANs do not have to take care of PAN configuration or dynamics, furthermore, they can use the uniform application programming interface (API) offered by the system. The middleware is responsible for connecting, disconnecting and rearranging user sessions.

An earlier version of MAIPAN is implemented in C on Linux. The lessons learnt from the implementation of this previous version are also incorporated in the design of the currently presented version of the middleware. A dispatcher application GUI is also available, which utilizes the API of the middleware's previous version [6]. In the future, beside implementing the current version, we plan to create some MAIPAN-enabled applications. Furthermore, we would like to analyze the efficiency of MAIPAN, as well.

References

[1] Mark Weiser: *„The Computer for the 21st Century"*, Scientific American, September 1991.

[2] M. Satyanarayanan: *„Pervasive Computing: Vision and Challenges"*, IEEE Personal Communications, August 2001.

[3] Cecilia Mascolo, Licia Capra and Wolfgang Emmerich: *„Middleware for Mobile Computing (A Survey)"*, In Advanced Lectures in Networking. Editors E. Gregori, G. Anastasi, S. Basagni. Springer. LNCS 2497. 2002

[4] Pubudu Chandrasiri, Ozgur Gurleyen, Yashar Shahabi, Christian Gehrmann, Annika Jonsson, Mats Näslund: *„Personal Security Domains"*, Contribution to the 10th WWRF Meeting, New York, October 27-28, 2003

[5] Rekesh John: *„UPnP, Jini and Salutation – A look at some popular coordination frameworks for future networked devices."*, California Software Laboratories Inc., Technical Report, June 17 1999

[6] MAIPAN's homepage: *„http://www.ronai.hu/maipan"*