Distributed Service Development in Personal Area Networks

Miklós Aurél Rónai, Kristóf Fodor, Gergely Biczók, Zoltán Turányi, and András Valkó

Ericsson Research, Traffic Lab, 1300 Budapest, 3., P.O. Box 107, Hungary,
Miklos.Ronai@ericsson.com

This paper presents the detailed description of the Middleware for Application Interconnection in Personal Area Networks (MAIPAN), which is designed to ease distributed service development for mobile and nomadic environment. This middleware provides a uniform computing environment for distributed applications that operate in dynamically changing personal area networks (PANs). MAIPAN hides the physical scatteredness and device configuration of the PAN and presents its capabilities as a single computer towards the applications. The solution provides easy set-up of PAN-wide applications utilizing multiple devices and allows transparent redirection of ongoing data flows when the configuration of the PAN changes. 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. Channels can be reconfigured when configuration or user needs change. In contrast to the approaches found in the literature, MAIPAN is a solution where session transfer, dynamic session management are tightly integrated with strong and intuitive access control security. A prototype implementation demonstrates the capabilities of the middleware.

1 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 describes in details the Middleware for Application Interconnection in Personal Area Networks (MAIPAN) 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 distributed 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 contains access control, flexible session management and transferable session control solutions. The middleware contains some intelligent functions, as well, which helps the user to control the PAN and improves human computer interaction (HCI). In the-

ory all kind of solutions for service discovery, physical, link or networking layers can be used with MAIPAN. However, currently MAIPAN is implemented on top of TCP/IP.

The paper is organized as follows. Section 2 is about related work; in Section 3 the basic concepts and the middleware's architecture are outlined. The API is described in 4 and the internal operation of the middleware (e.g., message exchanges, control functions and access control mechanisms) is detailed in Section 5. Finally, Section 6 concludes the paper.

2 Related work

Middleware are essential part of pervasive and mobile computing environments. Mascolo et al. in [1] discussed why traditional middleware (such as CORBA [2]) is not well suited for mobile environments and how a mobile computing middleware should be designed. Nowadays several projects are running in these research topics, some of them are presented in the followings.

The goal of the AURA project [3] is to provide each user with an invisible aura of computing and information services that persists regardless of location. The project Gaia [4] designs a middleware infrastructure to enable active spaces in which data and tasks are always accessible and are mapped dynamically to convenient resources present at the current location of the user. The Oxygen project [5] aims to develop very intelligent, user-friendly and easy-to-use mobile devices enabling users to communicate with the system naturally, using speech and gestures that describe their intent. In the frame of the Portolano project the one.world architecture [6] is designed, which is a comprehensive framework for building pervasive applications. The Cortex project [7] addresses the emergence of a new class of applications that operate independently of human control. The key objective of the EasyLiving [8] project is to create an intelligent home and work environment. The Speakeasy approach [9] focuses on the specification of minimal interfaces between devices using mobile agents and mobile code. The Virtual Device [10] concept considers all autonomous devices in the user's personal area network as one big virtual device having multiple input and output units, thus providing a coherent and surrounding interface to the user.

Similar to the Virtual Device concept, MAIPAN represents an entire personal area network as a single device to applications. On the other hand, MAIPAN represents a novel approach in its secure access control mechanism and the use of a transferable control role. MAIPAN access control ensures 1) seamless interworking of various devices of the same user, 2) protection of one user's devices from devices of another user, 3) still enabling controlled communication and lending between devices of different users. MAIPAN manages device access and configuration via a convenient central control entity, the dispatcher. MAIPAN is also unique in enabling the change of the dispatcher role, that is, the session control rights can be transferred between devices, from the old dispatcher to a new one.

The basic ideas of MAIPAN were introduced in [11] and the application programming interface is described in [12]. Now, in this paper we provide a detailed description of MAIPAN's internal operation, including its structure, session creation, dynamic session management, session control transfer and access control functions.

3 MAIPAN – Middleware for Application Interconnection in Personal Area Networks

3.1 Basic concepts and definitions

MAIPAN distinguishes among devices, applications and services. 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”.

MAIPAN is based on three concepts (Figure 1): pins, channels and sessions. 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.

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.

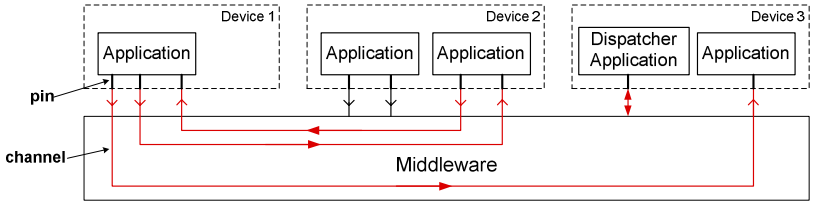

Figure 1: MAIPAN session

3.2 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 managers.

3.3 Transferring sessions

In the PAN at least one device playing the role of the controller entity is needed. In case this device disappears all concerned sessions will be automatically torn down. To keep up such session MAIPAN offers the possibility to transfer a running session from the current control entity to another one.

For example, to make some music in a meeting room one of the users creates an mp3 playing session. This way the user’s device becomes the control entity of the session. After a while, when the user wants to leave, she can transfer the session by telling to MAIPAN the identity of the new control entity, which can be for example another user’s PDA.

3.4 Reconfiguring sessions

In case a participant disappears (e.g., the user leaves the room, or the device’s battery is depleted), the concerned channels are automatically disconnected and the sessions have to be re-

configured. In the first step MAIPAN notifies the corresponding dispatcher(s) about the event. In the second step the dispatcher application(s) can decide which services to use instead of the disappeared ones. The dispatcher application can ask for user involvement, if there are multiple possibilities to replace the disappeared service(s), or it can decide on its own, if there is no or only one choice, or the user preferences are known. In the third step the dispatcher builds up the new channels or tears down the sessions concerned.

3.5 Architecture

Based on the concepts above we designed MAIPAN's architecture (see Figure 2). The aim of the *data plane* is to provide effective and secure data transport between applications, while the *control plane* is responsible for managing pins, channels, sessions and for handling security and access control.

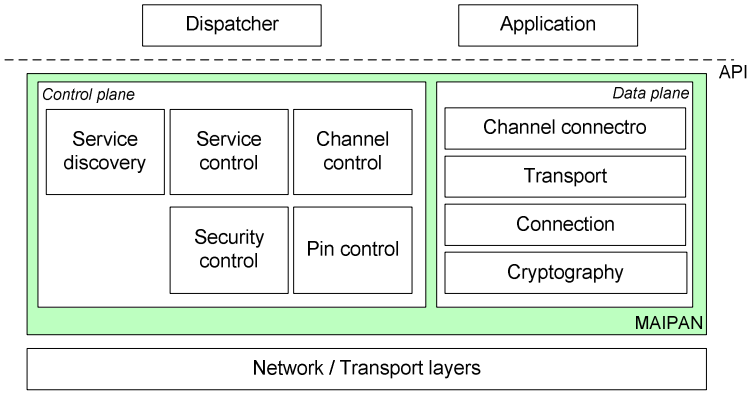

Figure 2: Architecture of MAIPAN

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., however these functions are not implemented, since the current version of the middleware is running over TCP/IP. 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.

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. 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 coordinates the authentication procedure between devices, manages the service access rights, and stores the necessary information for communication (e.g., security keys).

3.6 Implementation

MAIPAN is implemented in C on Linux [13]. In this implementation the applications are connected to the middleware stack via inter-process-communication (IPC) message queues, where each pin is represented by an IPC queue. When-ever an application sends data to another application, the middleware gets the data from the IPC message queue, creates packets and sends the packet to the remote end point of the pin's channel using UDP/IPv4 for transporting. Channels are built up according to the instructions of the dispatcher application, which gets information about available services in the PAN from a simple service discovery protocol (SDP). This simple SDP is based on broadcast messages, however in theory any kind of SDP implementation could be applied.

In order to demonstrate the operation of the middleware—beside implementing the dispatcher—we created a fileserver/client application and attached an mp3 player to the system [14]. This way we could set up a basic scenario, where the mp3 player could play an mp3 file located at a remote place.

4 APPLICATION PROGRAMMING INTERFACE

Application developers, who want to write PAN applications only and do not want to deal with PAN configuration, have to use just the service and pin registration functions of the API, which are described in the next section. Programmers, who are interested in configuring PANs and building dispatcher applications, can use all features of the API.

4.1 Service and pin registration

If an application wants to offer a service in the PAN, first it has to introduce the service to the middleware and register the input and output pins that are necessary for the use of the offered service. To do that the following message exchange is needed.

First the application sends a register service request message (Reg_Serv_Req) to the middleware. This message has to contain some service discovery protocol dependent information, which is automatically forwarded to the attached SDP. The middleware creates a new record for the service and a dedicated control channel for the application and returns an identifier, which will identify the service to the middleware in the future. Optionally the application can ask for a specific service identifier in the Reg_Serv_Req message. After this, the pins can be registered by sending a Reg_Pin_Req message for each pin. The message contains the following information: the type of the pin; whether the given pin is an input, an output or a bi-directional one; the minimum quality of service (QoS) parameters that the pin needs, such as reliability, reordering, flow control, encoding of transmitted data, etc.; optionally a specific identifier for the pin.

Inside the middleware the service control forwards each pin registration request to the pin control, where the information on pins is stored. Furthermore it sets default access rights for the pins by informing the security control about the new pins. In case of successful registration, the service control acknowledges each pin registration request and returns an identifier, which will identify the pin on the given device. New pins can be registered at any time, when an application decides to do so.

Pins can be disconnected from the middleware by a Revoke_Pin_Req message, which has to contain the identifier of the pin. To disconnect services the Revoke_Serv_Req message is used, containing the identifier of the service. Revoking a service will delete all of its pins.

4.2 Gaining information about sessions, services and pins

With the Session_Info_Req message the dispatcher can query which sessions are the middleware aware of. In answer the middleware returns information on all sessions that were initiated locally and on sessions where local pins are involved. Information about services and pins can be gained with the Service_Info_Req message. In the answer the middleware returns the locally registered services, the identifiers of the pins that belong to the services and the parameters of them. Moreover the middleware queries the SDP and forwards its answer to the application, as well.

4.3 Creating sessions

The dispatcher application can initiate the setup of sessions in the following way. First, it has to indicate that it wants to set up a session with the Create_Session_Req message. In the answer, the middleware sends an identifier for the session. After this, the dispatcher can start sending requests to interconnect pins with the Create_Channel_Req message. The control application sends as many requests as many channels it wants to establish in the given session. The message should contain the followings: the identifier of the session, the identifier of the device where the first pin is located, the identifier of the first pin, the identifier of the device where the second pin is located and the identifier of the second pin; and the quality of service (QoS) parameters that this connection needs.

Channels can be deleted with the Delete_Channel_Req message. The request contains the identifier of the session and the identifier of the channel that shall be deleted. To delete a session the Delete_Session_Req message can be used containing the identifier of the session.

4.4 Transferring sessions

The transfer of a session can be initiated by the current controller device with the Transfer_Session_Req message. The message shall contain the identifier of the session and the identifier of the new controller device. After a successful transfer the new controller will be responsible for the entire session.

4.5 Access control

The device, which is the owner of a service, can delegate its access decision right to other devices, by adding the MAIPAN ID of the device to the managers list of the given service. For this purpose the Add_Manager_Req message have to be used. As soon as the middleware receives the request to add a manager to a given service, it informs the chosen manager about the request. If the manager accepts the request, it will be added to the managers list of the service. To delete a manager the Del_Manager_Req message is used.

Each device has a white and a black lists about devices, that contain which devices are allowed and not allowed to communicate with. The white and black lists can be modified by the dispatcher application with the Set_Device_List_Req and Del_Device_List_Req messages, indicating whether the white or the black list has to be modified and which device has to be added or removed.

Also each service has a white and a black list about devices, which are allowed and which are not allowed to control the given service. The manager device stores these lists locally, so it is possible that there will be white and black lists for the same service both on the owner and on the manager, as well. The Add_Controller_Req or Del_Controller_Req messages can be sent to the middleware, indicating which service's list has to be changed and which device has to be added or removed.

4.6 Changing focus

The middleware provides facilities for connecting more pins to a given application's pin. Since data can flow only between two pins (by default towards the pin that was first connected to the source pin), similar to the X-Windows environment, the notion of "focus" arises, to determine which application receives the data flow. But unlike in X-Windows, mouse movements or the ALT+TAB key combination on the keyboard do not solve the problem, since there might be neither a mouse nor a keyboard that could be used for this purpose. In a PAN environment the concept of focus needs to be generalized and extended to handle such situations in an intuitive, ergonomic manner (e.g., placing a button on the Tetris-box, that switches between the monitors that are connected to the display output of the Tetris application). MAIPAN supports the focus change function, although, the solution of this ergonomic problem is out of scope here.

The dispatcher can initiate a focus change process with the Change_Pin_Focus_Req message. The request should contain the following information: the identifier of the session, the identifier of the device where the pin is located whose focus has to be changed, the identifier of the pin whose focus has to be changed, the identifier of the device where the pin is located which has to receive the focus and the identifier of the pin that has to receive the focus.

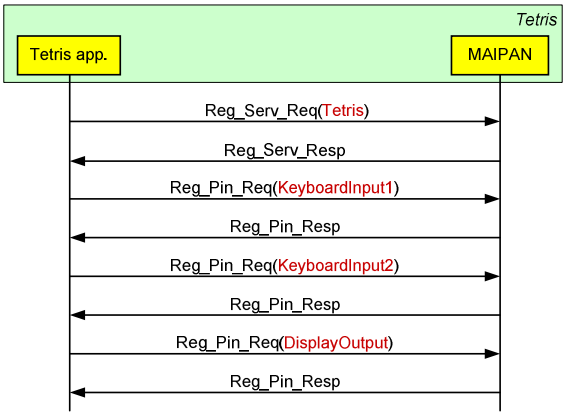

Figure 3: Registering the Tetris

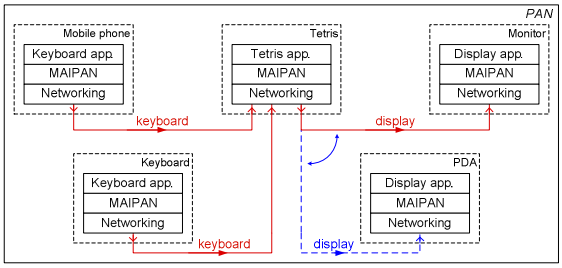

Figure 4: Channels in the Tetris session

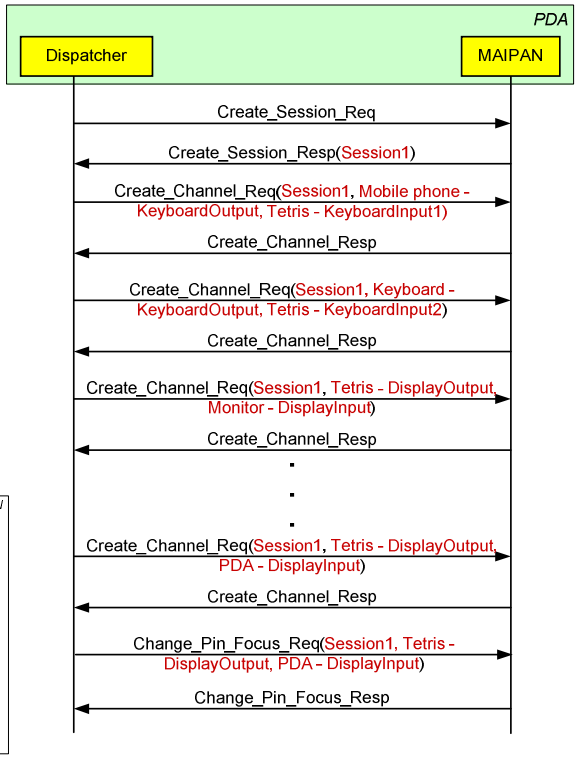

Figure 5: Creating the Tetris session and changing focus

4.7 An example on using the API

A MAIPAN scenario can be, for example, the installation of software purchased in a MAIPAN-aware box, which is realized by adding the box to the user's PAN (see Figure 3). 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. 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. Figure 3 shows the channels that have to be created in the Tetris session: 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.

Let's see how it can be realized with the API of MAIPAN. To run the Tetris service, first the Tetris application has to register its pins to MAIPAN: two keyboard inputs and one display output is needed for the application to function properly. The message sequence of registering the Tetris service to MAIPAN can be seen in Figure 4. The mobile phone, the keyboard and the display applications has to register their services in similar way. Note that not every parameter is shown in the message sequence charts of this example.

To use the Tetris game the dispatcher application running on the PDA has to interconnect the keyboard output pin with one of the keyboard input pins of the Tetris, the mobile phone's keyboard output pin with the other keyboard input pin of the Tetris and the Tetris display output pin with the display input pin of the monitor. This message sequence can be seen in Figure **5**. From now on the game can be used by the two users.

After a while, when the users leave the room where the monitor is located, the display output of the Tetris game is redirected to the PDA's display input. This is done with the focus change function of the middleware. This message sequence is also shown in Figure **5**.

5 Internal operation and detailed architecture

5.1 Addressing

Devices are distinguished using unique identifiers (MAIPAN ID). Devices also need a routing identifier (routing address), which is used by the routing layer to route the packets between devices (e.g., this identifier can be an IP address). The routing identifier may be changed in case network connections change, e.g., two PANs merge or some devices appear or disappear. In contrast, unique identifiers never change.

5.2 Channel-end-point (CHEP) identifier

Channels are represented by their end-points. The identification of a channel end-point (CHEP) is twofold: it is identified by the unique identifier of the device where the end-point is located and by a CHEP identifier, which is unique on the given device. Thus channels are defined by four identifiers: two MAIPAN IDs and two CHEP IDs.

5.3 Authentication of devices

In case two devices want to communicate with each other, they have to initiate a procedure to establish a trust relationship. The exact mechanism of the authentication procedure is out of this paper's scope. However, some ideas are described in the following. The authentication procedure can be started, for example, with the following:

- The user enters a code on her own device, which is printed on the device she wants to use. With this code the devices will be able to establish a trust relationship avoiding a man-in-the-middle attack.
- The user points with her device's infra red (IrDA) port to the device she wants to use, the devices exchange some messages via IrDA and create a secure key for further communication.

After this procedure the devices can communicate in a secure way and has to know the following about each other:

- They know each other's unique identifier (MAIPAN ID).
- They are aware of each other's human-readable device name, e.g., pda111, mouse123, keyboard456.
- They know the current routing identifier of each other (routing address).
- They have either a symmetric key or asymmetric cryptographic keys, that they use to secure their communication. Every node has a different key with every single node. The key(s) may have expiration time, in case they expire new keys have to be generated and exchanged.

The communication keys are stored in the communication secrets registry, which is handled by the cryptography layer of the data plane. Other security information (e.g., certificates) are stored in the device identities registry, which is handled by the security control part.

5.4 Access control

The service control part registers services and stores the information about them in the services registry. The service access rights are stored in the service access rights registry, which is handled by the security control part. Three levels of service access rights have been defined. The highest level is the owner, which is the device the service is running on. The owner decides whether a controller can access the given service. If the controller is welcome, then the owner puts its unique identifier into the controllers' white list of the given service. If the controller is not welcome, then its identifier is put in the controllers black list. The owner can delegate its decision right to the managers by adding the identifier of the manager to the managers list of the service. If the right is delegated, the manager stores a white and a black list per service locally, about controllers that can or cannot use the given service. The third level is the controller level. Devices that are in the controllers' white list can use the service to build up sessions; devices that are in the black list cannot use the service. These lists can be modified by the dispatcher application, and thus by the user. Managers and controllers cannot delegate their rights farther to other devices.

There are also a devices white list and a black list in the device access rights registry handled by the security control part, where the allowed and forbidden devices are listed. If a device is in the black list, then its communication establishment request is always refused. The dispatcher application running on the device can modify the content of these lists. The lists are handled by the security control part.

It depends on the operating system and on its administrator, who is allowed to run a service over the middleware. The access rights set by the operating system determine, whether a user can start a control application or a service. The middleware trusts the operating system (OS), so there is no explicit trust relationship defined between the middleware and the services running on the same OS with the middleware.

Every information on local pins—including the properties (e.g., minimum QoS requirements, direction) and the status (e.g., connected, free)—are stored in the pins registry, which is handled by the pin control part.

5.5 Session setup

A session is created, when all necessary pins are interconnected by channels. Only the control entity knows which channels form the session, thus other devices are not aware of this logical grouping of channels.

In case the channel control receives a request from a local dispatcher to connect two pins, it has to set up a corresponding channel. First, the channel control starts to create two channel endpoints at the participants by sending CREATE_CHEP_REQ messages to the pin controls of the

two devices (see Figure 6). Upon receiving such a request, the pin control has to check whether the controller is allowed to access the given pin. This is done by checking the access rights registry or by asking the manager of the service with an ACCESS_QUERY_REQ message.

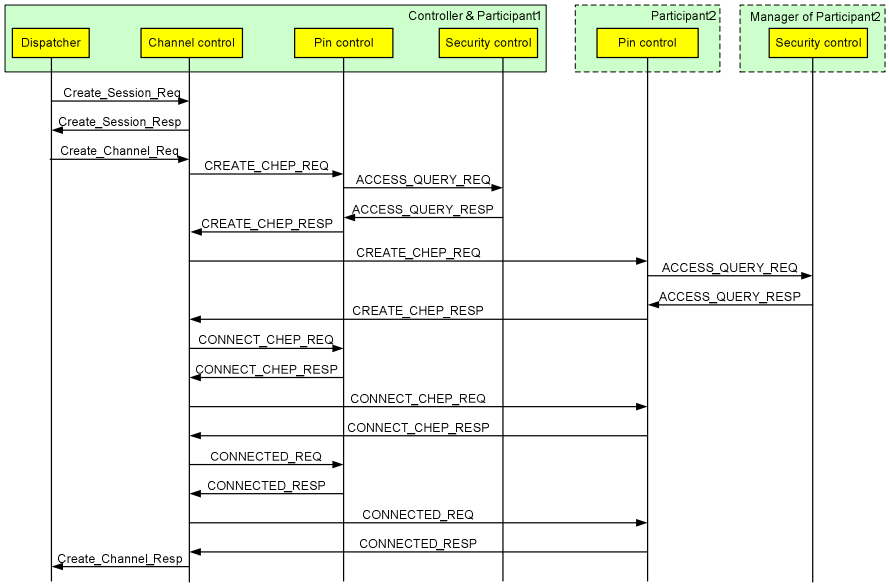

Figure 6: Creating a channel

In case access is granted for the controller to use the given pin, the pin controller checks whether the pin is free. Connection of a new channel is only possible if the pin is not in use by another controller, or if it is in use, but the current user resigns of its usage. In the latter case an ACCESS_QUERY_REQ message is sent to the corresponding controller that uses the pin in order to determine whether it wants to resign the usage. If the controller resigns, then the new controller can use the given pin and the new channel end-point can be created.

In the second phase of setting up a channel, the two newly created channel end-points are connected to each other. This is done by sending a CONNECT_CHEP_REQ message to both participants, which contains the corresponding remote end-point of the channel and the required quality of service parameters. Thus, both participants will be aware of the channel's other end. Finally, the setup of the channel is registered in the managed channels registry of the channel control, and a CONNECTED_REQ message is sent to both participants indicating that the set-up of the channel was successful, the data transmission can be started.

In case a channel has to be connected to a pin that is already in use by the controller itself, the channel is not set up, but is registered by the initiator's channel control part as a non-active channel, i.e., a channel that does not have focus on.

MAIPAN devices, which participate in the same session, and which have not exchanged data with each other for a long time, send alive messages to indicate that they are still there. The timer of sending the alive message is reset every time, when a packet is sent to the device the timer corresponds to.

5.6 Transferring sessions

Transferring a session is done by the channel control in two steps. In the first step the new controller has to be informed about the session that has to be transferred. Thus a TRANSFER_SESSION_REQ message is sent from the old controller to the new controller, which contains the channels that belong to the session that has to be transferred.

Channel control parts cannot decide whether to accept or reject a request about transferring a session, so they have to ask the local dispatcher application. This is done by using a special notification message (EVENT).

In the second step the new controller has to rebuild the concerned channels. This is done by setting up a new session at the new controller, as shown in Figure 6. However, in this case the concerned pins are in use by the old controller, thus for each ACCESS_QUERY_REQ the old controller gives a positive answer and resigns the usage of the given pin to the new controller.

5.7 Changing focus

Although, from an application point of view, several channels can be bound to a pin at the same time, in fact at a given time only that channel is set up, which has the focus. The focus change can be initiated only by that controller whose channel currently has the focus.

The first step of the focus change is to disconnect the pin that will lose the focus. This is done by a CONNECT_CHEP_REQ message sent to the device whose pin has to be disconnected. This time the message does not contain the identifier of the CHEP to connect to, thus the CHEP at the receiver node will not have any CHEP pair and so the CHEP at the receiver will be disconnected. The second step is to create a new channel to the pin that receives the focus with CONNECT_CHEP_REQ messages.

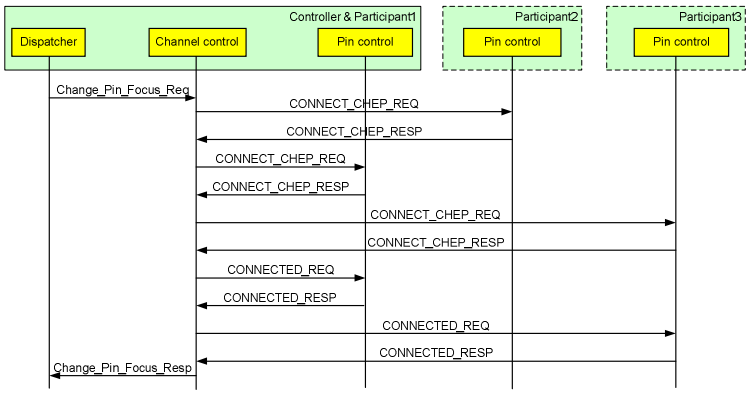

Figure 7: Changing focus

Figure 7 shows an example, where the focus is changed at Participant1; the pin at Participant1 will be connected to a pin at Participant3 instead of its current connection to the pin at Participant2. In this case the channel between Participant1 and Participant2 will be deleted and a new channel between Participant1 and Participant3 will be created.

6 Conclusion

In this paper we described the internal operation of 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. With MAIPAN distributed service development for mobile and nomadic environment can made easier.

MAIPAN represents a novel approach in its secure access control mechanism and the use of a central control entity. MAIPAN access control ensures 1) seamless interworking of various devices of the same user, 2) protection of one user's devices from devices of another user, 3) still enabling controlled communication and lending between devices of different users. MAIPAN manages device access and configuration via a convenient central control entity, the dispatcher.

MAIPAN is also unique in enabling the change of the dispatcher role, that is, the session control rights can be transferred between devices, from the old dispatcher to a new one.

According to MAIPAN's access control scheme, access to services is 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. Dynamic session management handles the situation when a participant disappears. In this case MAIPAN notifies the dispatcher application, which than tries to automatically reconfigure the session by involving new participants, or if there are multiple choices and the user preferences are not known, it may ask for user involvement. MAIPAN also provides session transfer, which is useful in the situation when the device that owns a running session has to leave. In this case, if the user wants to keep up the session after she left with her device, she can instruct MAIPAN via the dispatcher application to hand over the sessions from her device to another one.

In the future we plan to enhance the current implementation, create more applications running on top of the system and extend the architecture with various new functions to improve human computer interaction.

References

[1] 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

[2] A. Pope: "The CORBA Reference Guide : Understanding the Common Object Request Broker Architecture", Addison-Wesley, Jan. 1998.

[3] D. Garlan, D. P. Siewiorek, A. Smailagic, P. Steenkiste: "Aura: Toward Distraction-Free Pervasive Computing", IEEE Pervasive Computing, 2002., http://www-2.cs.cmu.edu/ aura/

[4] Gaia Project: "Active Spaces for Ubiquitous computing", http://gaia.cs.uiuc.edu/index.html

[5] "MIT Project Oxygen", Online Documentation, http://oxygen.lcs.mit.edu/publications/Oxygen.pdf

[6] Robert Grimm, Janet Davis, Eric Lemar, Adam MacBeth, Steven Swanson, Thomas Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, and David Wetherall: "System support for pervasive applications", ACM Transactions on Computer Systems, 22(4):421-486, November 2004.

[7] CORTEX Project: "CO-operating Real-time senTient objects: architecture and EXperimental evaluation"; http://cortex.di.fc.ul.pt/index.htm

[8] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, Steven Shafer: "EasyLiving: Technologies for Intelligent Environments", in Proc. of Handheld and Ubiquitous Computing Symposium, (Bristol, England), 2000.

[9] W. K. Edwards, M. W. Newman, J. Sedivy, T. Smith: "Challenge: Recombinant Computing and the Speakeasy Approach", MobiCom'02, September 23-28, 2002, Atlanta, Georgia, USA.

[10] Jonvik, T.E., Engelstad, P.E., Thanh, D.V.: "Dynamic PAN-Based Virtual Device", Proceedings of 2nd IASTED International Conference on Communications, Internet and Information Technology (CIIT'2003), 17-19 Nov 2003

[11] Miklós Aurél Rónai, Kristóf Fodor, Gergely Biczók, Zoltán Turányi, András Valkó: "MAIPAN: Middleware for Application Interconnection in Personal Area Networks", in proceedings of Mobiquitous 2005 conference, San Diego, CA, USA, July 17-21 2005

[12] Miklós Aurél Rónai, Kristóf Fodor, Gergely Biczók, Zoltán Turányi, András Valkó: "The MAIPAN Middleware", in proceedings of Mobilfunktagung, Osnabrück, Germany, July 17-18 2006

[13] Kristóf Fodor: "Implementation of a Protocol Stack for Personal Area Networks", Master's Thesis, Budapest University of Technology and Economics, June 2003

[14] Balázs Kovács: "Design and Implementation of Distributed Applications in Ad Hoc Network Environment", Master's Thesis, Budapest University of Technology and Economics, May 2003