Service Gateway Architecture for a Smart Home

Dimitar Valtchev and Ivailo Frankov, ProSyst Software AG

ABSTRACT

An implementation of the OSGi standard that enhances the standard and integrates many existing home protocols and home networks is presented. This enhancement extends the ideas behind the OSGi specification, allowing its direct application to real-life situations. Special emphasis is placed on realizing an effective and realistic gateway management system. The architecture presented was applied in several pilot projects.

INTRODUCTION

Recently the idea of a Smart Home has been the subject of many publications and conferences. A Smart Home is a house or living environment that contains the technology to allow devices and systems to be controlled automatically. The degree to which this control is exercised is variable, being a function of the cost, the home owner's preferences, and the type of building in which the technology is installed. A home that can automatically adjust the temperature and the level of security, and efficiently communicate with the outside world, is of obvious benefit provided it does not go too far by limiting the freedom of choice of the resident.

Implementing this concept is easier when using a gateway that provides features such as:
- Connecting the various home networks
- Connecting the home network and Internet
- Remote control and diagnostics of home devices
- A flexible mechanism for extending and updating the software
- A reliable and secure method of remote management

Various possibilities exist for the implementation of such a gateway. This article presents one approach. The approach is based on the work of the Open Service Gateway Initiative (OSGi) [1], one of the most promising specification activities in this field.

HOME NETWORKS AND RESIDENTIAL GATEWAYS

The complexity of a Smart Home solution lies in the variety of different protocols and media involved, and the requirements of the various services provided (Fig. 1).

The physical connections among devices can be implemented in various ways, mostly by using a wire or some type of radio signal (wireless). Because each system has different capabilities of speed, distance, and data volume, different physical media are suitable for different purposes.

The following physical media are currently in use:

Powerline. This refers to the electrical wiring that exists in the home to provide power to lights and appliances. One advantage is that there are already outlets throughout the home. Limitations as a network medium are low speed and consequently low data rates. These limitations are related to the bandwidth capacity of the wire. Currently powerline is used for applications with low data rate requirements, such as lighting and appliance networks, and some security applications. There are new technology developments that permit transfer of information and data over the powerline network at a much higher rate. The main communication protocols for power lines are X10 (www.x10.org), CEBus (www.cebus.org), LonWorks (www.echelon.com), HomePlug (www.homeplug.org), and EHS (www.ehsa.com).

Phone Line. In some countries most homes have several telephone outlets already in place. Phone lines can handle signals with very wide bandwidth, such as voice and data communication. Even entertainment networks can be implemented, using new technologies and applications developed recently. HomePNA (www.homepna.org) is the major phone line standard.

Wireless. There are several wireless solutions designed to be used in all types of networks, from traditional lighting, appliance, and security networks to communication, data, and some entertain-

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on November 10, 2008 at 15:03 from IEEE Xplore. Restrictions apply.

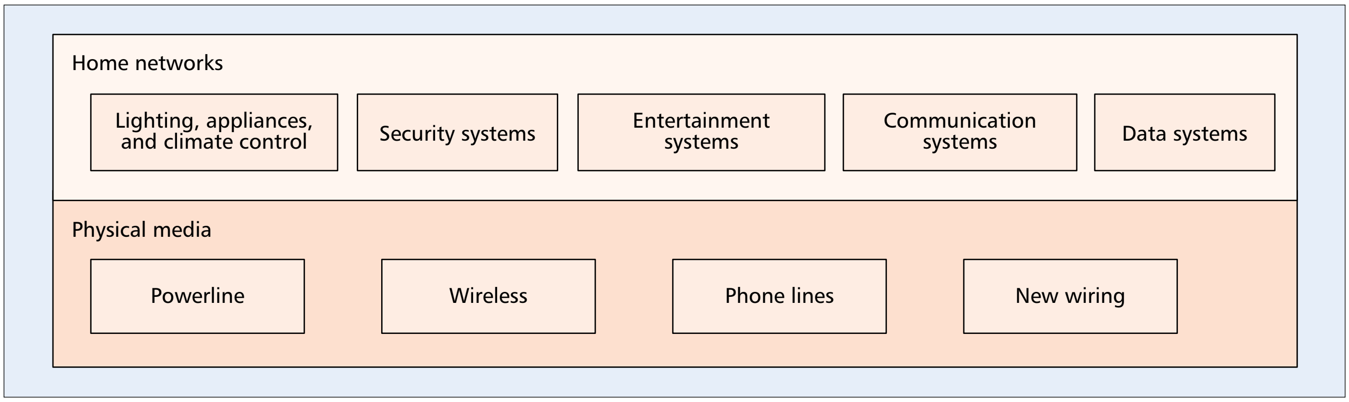

Figure 1. *Home network types.*

ment systems. Protocols in this group include Wireless LAN, HomeRF, Bluetooth, and IrDA [2].

New Wiring. "Structured wiring" systems installed specifically for data communications include RG-6 (coaxial) wire for multiroom entertainment systems; CAT 5 wire for data, audio, and video communications; and special wiring for speakers, home controls, and so on. Fiber optic cable is also being installed in some homes to provide extra future capacity. These types of wiring can be installed in new homes.

Popular communication standards that require dedicated wiring include Ethernet, FireWire (IEEE 1394), USB (www.usb.org), and EIB (www.eiba.org).

Even this short overview illustrates the large variety of existing protocols. No single protocol is suitable for all purposes because of the different physical characteristics and cost requirements.

On top of the physical media and low-level protocols, various types of home networks exist. Some of the protocols cover several network layers, even including the application layer.

Lighting, Appliances, and Climate Control: This is the traditional home automation and control network. The most widely used examples are EIB, LonWorks, X10, and CEBus.

There is an attempt to combine the most popular European standards (EIB, EHS, and BatiBus) in one network called Konnex (www.ehsa.com).

Security Systems: This includes traditional security components, plus video monitoring and other recent enhancements. The medium used is usually structured wiring, but sometimes powerline and wireless media are used.

Entertainment Systems: Audio, video, and theatre equipment can be located throughout the home.

The most promising protocols include HAVi (www.havi.org) and MHP (www.mhp.org).

Communication Systems: Telephone and intercom.

Data Systems: Multiple PCs and other data devices, most of which run over TCP/IP networks.

It is clear that a state-of-the-art Smart Home should integrate all these networks. Features such as remote control, diagnostics, and management require the availability of an Internet connection. Services like video on demand and interactive TV require a high-speed Internet connection.

A natural solution for integrating these networks is a central node that serves as a bridge between various local networks and the Internet. This is exactly the purpose of a residential gateway.

If the structure of the home network is known in advance, a simple gateway with predefined architecture can be used. For example, there are many integrated services digital network (ISDN) boxes that allow several telephones, fax devices, and PCs to be connected together. Unfortunately, it is not possible to know in advance what number and types of devices will be used in a private home.

To solve the problem, there is a need for the following:

- Special discovery protocols, such as Jini (www.jini.org), UPnP (www.upnp.org), and EHS to manage the integration of new devices into the network
- A mechanism for dynamic update of the software on the gateway

In order to support the ability to install and replace software modules on the gateway dynamically, the software must be designed as separate functional blocks that interact with each other. These functional blocks may be called *services*, and a gateway with software structured as separate functional blocks is called a *service gateway*.

There are several existing approaches to creating a standard execution environment for services. Examples of such standards are OSGi, Microsoft.NET, HAVI, and MHP.

OSGi is one of the most convincing candidates as "the standard," due to the facts that it is already quite mature and allows easily constructed bridges to all other existing standards.

OSGi Service Platform

The OSGi Service Platform is a general-purpose, secure, managed Java software framework that supports the deployment of extensible and downloadable service applications known as *bundles* [1, 3].

The OSGi-compliant gateway can download and install bundles, when they are needed and uninstall them when they are no longer needed. Modules may be installed and uninstalled "on the fly," without need to restart the whole system.

This allows the creation of very flexible architectures. The OSGi architecture enables the following:

- Remote control and diagnostics

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on November 10, 2008 at 15:03 from IEEE Xplore. Restrictions apply.

The concept of a service gateway presupposes a high degree of user friendliness. The end user is allowed to make decisions at the service subscription level, rather than having to worry about details such as version of the operating system, which revisions of drivers are needed, and so on.

- Dynamic software update (i.e., life cycle management)
- Remote management
- Building systems that are open to third-party software

There are various fields in which service gateways with such features are very useful:
- Home
- Office and building automation
- Automotive
- Industrial automation
- Medical care

The current version of the OSGi specification is 2.0. Besides the framework, the specification includes several other basic services such as HTTP service, log service, configuration management, permission administration, preferences, user manager, and device manager. For more details on the OSGi standard see [1].

The specifications of these services do not cover all scenarios in a real home network. Consider the example of connecting a new device to the home network. The device manager is normally responsible for finding and downloading a driver for the new device, but the exact procedure depends strongly on the protocol used.

When a new device is installed in the home network, it must be detected. There are two ways to handle detection of a new device:
- A discovery protocol
- A tool for manual configuration of the gateway

When a device is detected, a device driver has to be installed on the gateway through a device manager. The drivers can be stored either on the gateway or somewhere in the Internet.

It is possible to download a file directly from an arbitrary site on the Internet, but only in the case when a direct connection to the Internet is enabled. However, this leads to problems with security. The best solution is to download the driver from the gateway operator, because it is responsible for the security of the whole system.

If a protocol with discovery features is used the device will announce its presence after being connected to the home network. The gateway detects the device, assigns a device ID, and through the device manager checks whether there exist suitable device drivers on the gateway. If not, the gateway connects to the gateway operator and searches for an appropriate driver.

If a suitable device driver cannot be found, the gateway installs a driver with generic features in order to provide basic control capabilities.

If a new device (e.g., a washing machine) uses a protocol that does not have discovery features, then the user can start an "add new device" wizard and tell the system that a washing machine with specified ID has been added to the network, the type of washing machine, and where the washing machine is located. The procedure after this point is the same as for a washing machine using a discovery protocol. Obviously the scenario with the discovery protocol is more attractive, which is why protocols like EHS and Konnex are very promising.

Currently there is intensive work going on inside OSGi in several other areas. One of the tasks with highest priority is to define a reliable and convenient remote management architecture.

The concept of a service gateway presupposes a high degree of user friendliness. The end user is allowed to make decisions at the service subscription level, rather than having to worry about details such as version of the operating system, which revisions of drivers are needed, conflicting dependencies among different software packages, and so on. The service gateway is also expected to have high availability and to deliver services with a specified level of determinism. These requirements lead to a need for management and monitoring of the gateway. The requirement to manage and monitor the gateway is not user friendly, and should be a service provided by the gateway operator. However, physical visits to users for administration, installation, and troubleshooting purposes are extremely expensive and inefficient.

For these reasons, it is very important that the gateway operator be able to administer service gateways remotely. Servicing a large number of users requires significantly more: it is necessary for the gateway operator to be able to administer many service gateways at a time.

The group of service gateways to administer may be physically located in widespread regions and run on different devices, operating systems, and virtual machines. Furthermore, users have differing needs and varying service bundles installed. This calls for administration and installation tasks to be performed in an intelligent manner, with the users serviced limited to those who need them or request help.

Another important issue is the possibility for sales or distribution of additional services by the gateway operator or an external service provider. A system needs to be organized whereby the end user may review services available on the market, purchase them, and have them correctly installed on his/her service gateway. In addition to the challenges of remote administration, this requires the resolution of a number of security issues. These issues include protection of the end user's privacy, processing of payments in a secure manner, and protection of the user from malicious interference.

A remote management specification that addresses these and other important issues is still lacking. The prompt completion of this specification is crucial for the success of the OSGi concept.

By offering a flexible and open architecture and defining some of the basic services, the OSGi provides a good starting point for implementing the residential gateway. A complete Smart Home solution, however, requires that many additional services be defined and developed by OSGi vendors, and such services naturally must meet the OSGi specification.

ADD-ONS TO APPLY OSGI IN A SMART HOME

We have designed and implemented additional components that provide an end-to-end solution [4]. These are:
- A set of system components that can be used as basic bundles for developing diverse applications

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on November 10, 2008 at 15:03 from IEEE Xplore. Restrictions apply.

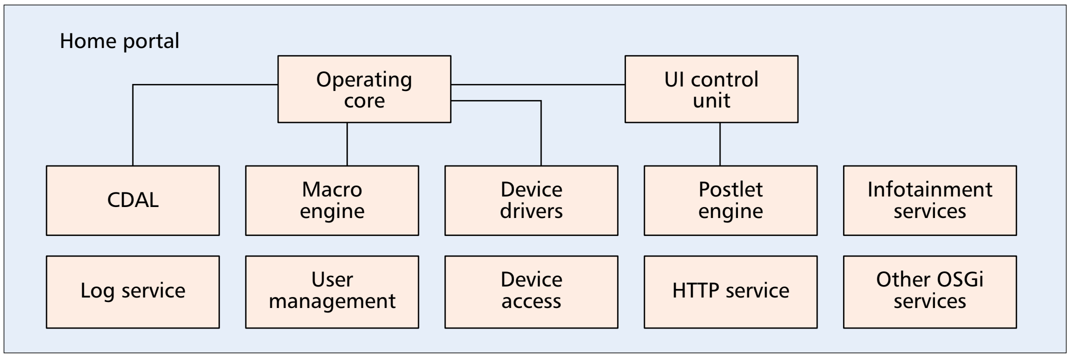

Figure 2. *Components of the home portal.*

One important design requirement of the macro engine was that it should provide as simple an interface as possible, so it could be used by the ordinary customer who is not necessarily an IT specialist. At the same time it had to be flexible, and not so simplified as to become useless.

- Home portal extension
- Remote management system

These three value-added groups of components will be presented in the following sections.

SYSTEM COMPONENTS

Most of the system components fit into one of the following groups:

- Device access — provides support for various hardware protocols.
- User interaction — supports user interaction with the gateways through HTML, WAP (www.wapforum.org), and SMS. Services like HTML and WAP browsers belong to this group.
- Internet access — Internet services like POP3, FTP, NNTP, and SMTP (www.ietf.org).
- Network management applications — provides network-related services like DHCP (www.ietf.org), dialup, firewall.
- Management protocols — implementation of SNMP (www.ietf.org).
- Message/RPC protocols — used for remote method invocation and administration of the framework. The message-based protocol for synchronous and asynchronous communication provides a secure and reliable mechanism for data exchange among devices, using various transports such as TCP/IP, SMS, or RS232.
- Security support — implementation of protocols like SSL (www.netscape.com), TLS (www.ietf.org), and WTLS (www.wapforum.org).

AUTOMATION AND MULTIMEDIA SERVICES: HOME PORTAL EXTENSIONS

The home portal has a pluggable architecture that uses postlets. Postlets are a proprietary extension of servlets that provide the capability to automatically integrate a servlet into the home portal. The postlet automatically incorporates all components of a given group that appear in the framework. This allows easy customization and further expansion with custom-built components to suit customer needs.

Common Device Abstraction Layer — The home portal is based on a protocol-independent device application programming interface (API) called Common Device Abstraction Layer (CDAL). All services provided by the home portal work through the CDA. This allows them to work with any kind of device — UPnP, Lonworks, Bluetooth, Jini, EIB, EHS, X10, HAVi, CEBus — without need for customization.

Macro Engine — Another vital part of the architecture is the macro engine. This provides a capability for programming a sequence of operations on devices that is automatically triggered by a compound triggering condition. One important design requirement of the macro engine was that it should provide as simple an interface as possible, so it could be used by the ordinary customer who is not necessarily an IT specialist. At the same time it had to be flexible, and not so simplified as to become useless.

The basic components of the home portal are presented in Fig. 2. They interact with each other using the mechanism provided by the OSGi framework. In this figure only connections important for understanding the main dependencies are shown.

The home portal consists of the following components as a minimum:

- A component for direct monitoring and manipulation of devices. This component is based on CDAL, which makes it protocol-independent. It allows all kinds of devices to be displayed, rather than the home portal display being limited to one type of device such as X10.
- A component for macro sequences (macro engine). This component makes it possible to write, delete, and modify macros, activate and deactivate them, and run them manually without waiting for triggering events. A library of ready events can be provided for direct use, for example, "all lights on," "all lights off," "all units off," "wake up with music," and "random switch on/off" (to simulate the house being occupied when the resident is away).
- Components providing various infotainment services: a mailbox, weather information, news, calendar with schedule, streaming video, mp3 player, photo album.
- Specific components that are distributed with a specific device and provide more detailed custom control over this device or a customized visual interface.
- A component for user-friendly device selection, using a customized map of the house rather than lists of devices.

More components can be added according to need. All components have the same look and feel, and a common operating logic.

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on November 10, 2008 at 15:03 from IEEE Xplore. Restrictions apply.

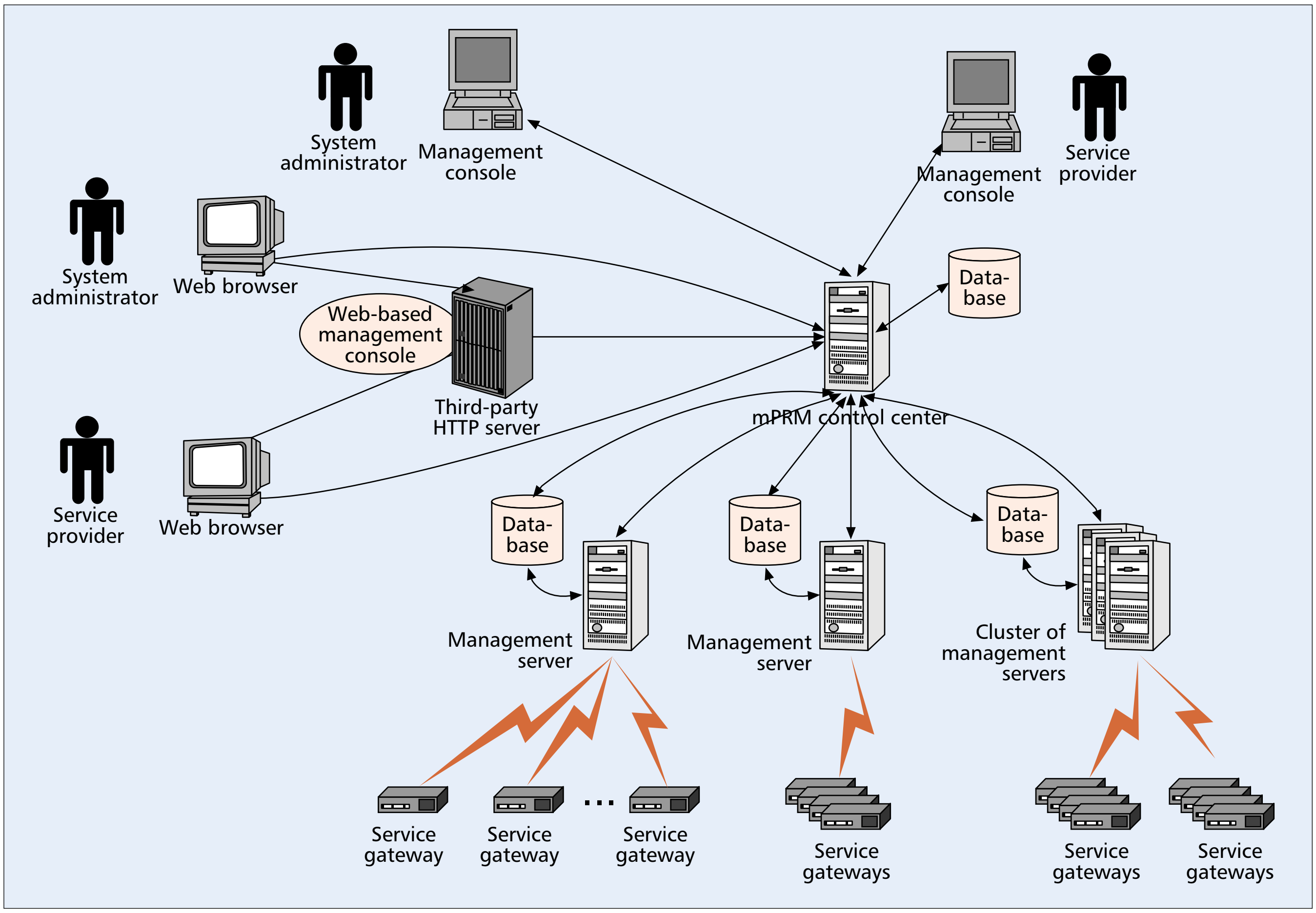

Figure 3. *mPRM architecture.*

There is a clear separation between the UI control unit (visual part of the system) and the operating core (macro engine, device drivers). This decoupling makes the UI control unit an independent part of the system. As a result, it can reside on either the same framework as the core or a remote system, communicating with the core. In this way the main framework, usually running on an embedded system with limited resources, can be relieved of the burden of the visual administration components, since they are not needed for the operation of the system.

The Remote Management Concept

The concept of remote management is implemented in the mPower Remote Manager (mPRM). An mPRM system contains the following modules (Fig. 3):
- One control center
- One or more management server(s)
- A Web server with servlet engine (optional)
- One management console
- Database server (LDAP or SQL)
- One or several OSGi service gateways

mPRM Control Center — The control center module provides a central point of control for the entire system. It acts as the mPRM interface to the external world, containing system administrators, system providers, external management systems, and so on. Its function is to enable system administrators to perform system management, and to allow service providers to perform service publishing and management. The control center can be accessed via diverse interfaces, including GUI, API, and various network protocols.

Management Server — A management server handles communication with service gateways. It monitors changes in gateway configuration, schedules management jobs for execution, and keeps track of job results reported by service gateways.

A management server uses a database (LDAP directory database or SQL RDBMS) for storing configuration information about all gateways under management, as well as information on management jobs pending execution. A management server is responsible for continually synchronizing the configuration of service gateways with configuration data stored in the database.

An mPRM system can have multiple management servers, each managing its own subset of gateways. The system administrator uses the control center to statically define a gateway subset as a subtree in the database. This approach has two benefits. First, it allows multiple management servers to share a large load. Second, the manager can be placed closer to the specified subset of gateways, in terms of network topology and data bandwidth, ensuring faster data exchange.

Each management server can store data regarding the gateways it handles on a separate and independent database server, although the system can still be configured with a single serv-

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on November 10, 2008 at 15:03 from IEEE Xplore. Restrictions apply.

er. This avoids the possibility of the database becoming a bottleneck in a complex heavily loaded system involving large numbers of service gateways and management servers. Another advantage is that if the system is distributed over a WAN and the connections among some of the components are relatively slow, every management server can be connected to its database via a faster LAN connection.

Management servers can also be grouped into clusters. A management server cluster is logically equivalent to a single management server: all management servers in the cluster manage the same subset of gateways and use the same database server. The cluster allows "intelligent" dynamic load balancing of active service gateway connections, and adds fault tolerance capabilities to mPRM.

Management Console — This provides a GUI, making it convenient for administrators to perform system management tasks such as scheduling management operations on service gateways, monitoring gateway status, and configuring the mPRM system itself. The management console has two variants: as a standalone graphical application and as a Web-based application used with a Web browser. Web-based applications can be hosted on a third-party industry-proven Web server for better performance and scalability, or accessed directly from the control center host.

The management console can be used by service providers and service aggregators to publish and manage services available for service gateway end users. The console also provides the ability to manage service subscriptions, monitor service usage, and generate billing reports.

Database Server — This section refers to a "database" as a logical set of related data, and "database server" as a physical machine which hosts one or more databases. mPRM uses several databases for storing several types of data:

- One database per management server that stores data about service gateways controlled by this management server, their current configuration, pending management jobs, and so on
- Database for storing users, user roles, service subscriptions, and preferences
- Database for storing bundle information, service packages and service configurations
- Database for storing payment data: billing information, currency exchange rates, and so on
- Database for storing public key identifier (PKI) information, such as X509 certificates (www.itu.int)
- Database for storing configuration information for mPRM itself: management server, database servers, and so on

These databases can be managed by one or more database servers. One advantage of distributing gateway configuration information on more than one database server is explained above. Another advantage is the possibility to use different database types for different databases. For example, it is common to place the user, user preference, and PKI information in directory databases so that mPRM can access this type of information through LDAP in an external directory server. The databases can be initialized from existing information already created by other applications. Other kinds of data, such as service gateway configuration, can be stored in an SQL RDBMS.

As mentioned earlier, if using more than one database server is not desired, mPRM can be configured to store all data on one.

Communication Protocol — The protocol used for communication between the remote manager and the gateways is crucial for the stable and effective work of the entire management system. The protocol must be reliable, secure, and capable of handling thousands of simultaneous connections.

HTTP is one possible choice for communication protocol. Advantages are that it is widely used, secure (with SSL), and functions through firewalls and proxies. mPRM can use HTTP as the transport protocol (HTTPT).

In many cases, however, HTTP has a great disadvantage: it causes a big load on the server site, since a new socket is opened for each connection. In order to solve this problem and provide better load balancing, we developed a *UDP-based transport protocol* (UDPT).

UDPT is an extension to UDP. It is a packet-based protocol that supports virtual communication sessions. It provides mechanisms for ensuring reliability and security encryption (note that standard UDP is unreliable and unsecure). Reliability is ensured by the use of unique packet identifiers, acknowledgments, removal of duplicates, and retransmissions.

The following list summarizes the main features of UDPT:

- Ensures reliability over UDP through the use of unique packet identifiers, acknowledgments, duplicate removal, and retransmissions.
- Packet orientation.
- Provides mechanisms to minimize the number of transmitted packets. This is especially useful when the gateway uses a low-bandwidth physical medium.
- Provides both synchronous and asynchronous message exchanges. Asynchronous transmission can be either reliable or unreliable as options.
- Provides a virtual communication session between endpoints. The gateway or management server can initiate the session.
- UDPT is extensible. This is needed at minimum for providing security encryption on top of UDPT.
- Provides support for sending and receiving long messages by separating them into smaller UDP packets, then concatenating back together on the remote host.
- Time-dependent parameters such as packet acknowledgment timeout, ping interval, session timeout interval, and hold on interval are dynamically tuned, depending on current load (current number of active sessions) for best performance and reliability.

The main shortcoming of the UDPT-based protocol is that it needs additional adjustment to the firewall (if one exists, which is the normal case) in order to allow connections to the UDP port.

Management servers can also be grouped into clusters. A management server cluster is logically equivalent to a single management server: all management servers in the cluster manage the same subset of gateways and use the same database server.

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on November 10, 2008 at 15:03 from IEEE Xplore. Restrictions apply.

OSGi presents a very practical, attractive approach for the realization of the Smart Home concept. The open specification and technology support for this standard make its wide acceptance highly probable.

We have created a test to determine the maximum number of gateways that can be connected simultaneously to a management server. This test simulates network communication using the mPRM network protocols employed for gateway-to-management server communication. The goal was to attain overload conditions. The test was designed so that a configurable number of gateway clients, normally between 500 and 1000, can be simulated on a single physical machine. This provides the opportunity to explore the behavior of mPRM with thousands of gateways connected to the system. Under these test conditions, using UDPT allows the management of 10 to 20 times more gateways than using a TCP/IP-based protocol such as HTTP.

An integration of mPRM with existing third-party J2EE application servers is also provided. It is based on the Java 2 Enterprise Edition (J2EE) connector architecture and has the following characteristics:

- Provides mPRM functions to applications running on a J2EE server. This includes a set of components that can be deployed on a J2EE-compatible application server and provides an interface to resources managed by mPRM: remote gateways, bundles, service packages, service subscriptions, and so on.
- Provides support for easier development of applications distributed between service gateways and back-end application servers. This means providing a communication mechanism for exchanging data between application logic residing on a service gateway and a server side application logic running on an application server. This mechanism must be natural for J2EE applications, but should not overburden the service gateway with heavy J2EE class libraries.
- Provides a mechanism for delivery of content (static or dynamic). This can be applied not only to content provided by an application server, but as an interface to other content providing systems: Web servers, media streaming servers, and so on.

Conclusion

OSGi presents a very practical, attractive approach for the realization of the Smart Home concept. The open specification and technology support for this standard make its wide acceptance highly probable.

The current version of the specification requires that vendors define and implement additional OSGi-compatible services in order to build an end-to-end solution.

We have described an implementation of an OSGi-compatible solution that integrates many existing home protocols and standards. This solution extends the ideas behind the OSGi specification, allowing its direct and successful implementation in many real-life applications.

References

[1] D. Marples and P. Kriens, "The Open Services Gateway Initiative: An Introductory Overview," *IEEE Commun. Mag.*, Dec. 2001, pp 110–14.

[2] D. Setty, "Which Network in Home — Linking the Standards Together?" *Home Net. Conf.*, London, U.K., Nov. 2000.

[3] OSGi Specification v. 2.0, Nov.2001; http://www.osgi.org

[4] ProSyst product documentation: http://www.prosyst.com

Biographies

DIMITAR VALTCHEV (d.valtchev@prosyst.com) is a chief technology officer of ProSyst Software. His main research interests are in the fields of nonlinear phenomena, signal processing, distributed computing, and embedded systems. He holds an M.Sc. in electronics, an M.Sc. in applied mathematics and computer science, and a Ph.D. in electrical engineering from the Technical University of Sofia.

IVAYLO FRANKOV (i.frankov@prosyst.com) is an application engineer at ProSyst Software. He has taken part in several home and building automation projects as lead engineer. He is interested in various areas of automation technologies. He received his M.Sc. in electrical engineering and electronics from the University of Gabrovo.

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on November 10, 2008 at 15:03 from IEEE Xplore. Restrictions apply.