https://doi.org/10.26552/com.C.2016.1A.4-12

Jakub Hrabovsky - Pavel Segec - Peter Paluch - Marek Moravcik - Jozef Papan *

USABILITY OF THE SIP PROTOCOL WITHIN SMART HOME SOLUTIONS

A smart home is being an important part of research due to its impact on the life of many people, who decide to try this partially known concept of building and controlling their houses. In this paper we present a little different approach to the implementation of a smart home structure and its methods in real environment. As a part of the work we specify main requirements on the architectural design, which should be considered in final solution. Our research is focused primarily on the use of open standards and tools. This approach became popular because it provides an alternative view in comparison to commercial ones, where mostly proprietary and publicly unknown protocols are used. According to this, three theoretical architectures are proposed and presented with their pros and cons. In the paper we concentrate mainly on the usage of SIP protocol and issues related to the communication between a remote user and a smart home. We are proposing two logical communication models. Each model describes the type of used SIP messages and their message flows. In addition, we consider financial aspects of simple and illustrative smart home solution, which as well pose issues to end users. Our solution offers implementation with minimal operational costs and high facility control of the smart home. The paper also includes a survey of potentially usable protocols for smart homes and the list of software (SW) and hardware (HW) components, which have been used to build a testing environment.

Keywords: *Home automation, smart home, SIP, session initiation protocol, central management.*

1. Introduction

The development of new technologies influences almost all fields of human life including home environment. By using new inventions and communication standards, we are able to automate operation of whole buildings, control their parts and achieve their optimal usage. The optimization causes a huge reduction of operational costs and time needed for flawless run of automated buildings.

An example of intelligent building is the smart home, where all core functions are controlled by special entity that is manageable by end users. Common functions are heating, lighting control, security and functions related to danger detection. The term “smart home” comes from the idea of automating actions of common house devices and interconnecting them to be capable of co-operation without user interaction.

The smart home has to be seen as a functional cluster of subsystems, which correspond with mentioned functions. Each subsystem is required to make its specified functionality and co-operate with other subsystems. They all together make one complex system.

In the heart of the system is the main controller, known as the central control unit. End users, using for example their mobile devices, are able to manage controller activities. Users send commands to the controller and as an answer they receive requested information about actual situation in their home. With this approach they are able to remotely access their houses independent of their actual place and time of day. The controller is also responsible for the management and interaction of each subsystem inside the smart home. This central control unit provides unified control of all electronic devices, which are members of the smart home. To ensure this, usually a special purpose software program is running inside the central control unit. It receives data from all connected devices, processes them and performs appropriate actions depending on actual situation.

There are other devices beside controller, which play a role within an architecture. They build a group where they are all interconnected via a special network. This way they are able to exchange information about different parts of home environment and so create its overall view available for end users. This service together with the remote access highly increases comfort of end users.

* **Jakub Hrabovsky, Pavel Segec, Peter Paluch, Marek Moravcik, Jozef Papan**
Faculty of Management Science and Informatics, University of Zilina, Slovakia
E-mail: jakub.hrabovsky@kis.fri.uniza.sk

In our proposal we are relying only on open communication standards, mainly the Session Initiation Protocol (SIP) [1]. This approach is different in comparison to methods used in commercial sphere, where proprietary and publicly unknown protocols are mostly used. Our approach brings another usage proposal of known protocols, SIP in this case. The choice of common and well-defined protocols brings us considerable advantages. The primary advantage allows us to design clear and relatively simple architectural model. As the second advantage, it provides us with good accessibility of all elements needed for its realization, such as HW and SW entities.

When we created an architectural model, we were considering the system autonomy, comfort of end users, simplicity and minimal costs as the main model parameters. In the paper we propose three slightly different design architectures and a solution based on one of them to achieve considered properties.

The rest of the paper presents related works, analysis of components for building a smart home, as well as proposed architectures and communication models. Brief overview of reference research papers is introduced in section 2. Section 3 gives the description and tasks of each logical entity included in the smart home. The section also presents four potential communication protocol candidates, which can be used in environment of the smart home. Section 4 describes our three architecture proposals, which utilize the SIP protocol. The section also includes description of two logical communication models, used messages and theirs flows. Section 5 presents a particular proof concept implementation with a list and short description of used hardware devices and software tools. Results and experiences coming from the realization and subsequent conclusion with proposals for the future work are presented in section 6.

2. Related work

There are several researches done in the field of the smart home development and the usage of common open protocols [2] presents the use of SIP and the propagation of extended location information gathered from various sensors (wired and wireless) via presence service as a context for dynamic behavior of home devices. The paper emphasizes important ability of end users to have full control over the home and its customization in the form of various profiles. A uniform model of heterogeneous entities of the smart home, which are using a common SIP interface with extensions as an abstraction layer, is described in [3]. The author specifies three basic communication forms (commands, events and sessions) with their particular realization via SIP interface. At the same time, author provides an abstract interface between a high-level programming (applications) and a low-level communication (SIP) defined in an architecture description language (DiaSpec). DiaSpec hides details of used technology and simplifies creation of new applications. To take advantage of open protocols like SIP, some approaches are using SIP infrastructures and applications already in existence to implement a smart home model. Such approach is presented in [4], where existing IP Multimedia Subsystem (IMS) provides SIP services (registration, session management, instant messaging and presence [5]) for an instance of the smart home [6] and [7]. Such approach allows a researcher to focus more on the design of a signaling gateway between SIP and hardware communication protocols like KNX. In [8] is provided an example of signaling gateway implementation deployed as a SIP adapter. As the gateway platform an ATmega board with an Ethernet shield is used. The platform is programmed in C that uses functions of the oSIP library. Another example of such gateway and its implementation to Raspberry Pi is described in [9]. Authors propose the usage of summarized SIP Uniform Resource Identifier (URI) of the smart home in comparison with the usage of several separated SIP URIs, each for individual home device. The main focus of the paper is placed on the security and the access control for home private data, where the concept of security layers is proposed. In our approach, we follow results and principles identified in mentioned papers, mainly the last one. We provide our own solution driven by the analysis described in next sections.

3. Structural view of the smart home

The smart home consists of elements that perform various functions - central control unit, sensors, control elements and remote mobile clients. We also consider a communication bus as a separate part of home structure. Here we provide a brief description of each entity and its role. It serves as the reference of architectural design.

A. *Central Control Unit*

Central control unit (CCU) is the main part of smart home. It controls other subsystem parts and also serves as a gateway between a remote client and an internal system. It achieves this by receiving and collecting data from home sensors, processing them to gain some valuable information and depending on the results by executing appropriate actions. Actions can be classified into two groups. Classification of an action depends on its initiator, which provides stimulus. If the initiator is the controller itself, we talk about scheduled actions. Otherwise (initiator is another device, i.e. one of sensors or remote users) the action is identified as executed. Two standard actions are sending information about a specific subsystem to the remote client and the modification of parameters of specific subsystem. In this paper the term home gateway (HG) is used to identify CCU.

B. *Sensors and control elements*

Sensors are sensible devices, which monitor a specific subsystem of the home environment. They extract appropriate data from the environment and send them to HG. Control elements, known as actors, are responsible for making desired changes in the behavior of home appliances. They are based on commands received directly from HG. Sensors and actors create the lowest layer of the system.

C. Remote mobile client

As a remote client is usually used a mobile device (Smartphone or tablet) with a software client. User uses the client to send requests (commands) to HG and eventually receive responses (depend on the type of request). The client serves as a user interface for remote access to HG. Software client has to be appropriate protocol application. This depends on a protocol, used for a message exchange between the client and HG. More information about our application is provided in section 5.

D. Communication bus

The communication bus is very important because it connects all subsystems and their elements together into one system. The format of communication via the bus is defined by used communication protocols. In our solution, we need to handle two independent communications applied at two separate places of the system because of their distinct requirements on network parameters. Main parameters that have to primarily be taken into account when choosing a right protocol are speed, distance, security and power-consumption.

1) The communication between HG, sensors and control elements

The communication relates to the lowest layer. Communication protocol focuses on the hardware connection with simple data exchange, where data format has the form of signals with two possible levels (on/off). Protocol is responsible for the interconnection of intern environment of the smart home – sensors and control elements. Therefore, it puts emphasis on power-consumption aspects. For completeness, examples of open protocols suitable for this type of communication are X10, ZigBee Home Automation, Z-Wave, Insteon, Bluetooth Low Energy (BLE), Wi-Fi and Near Field Communication (NFC). Wired or wireless technology is used as a transfer medium and depends on the specific protocol and its support.

2) The communication between HG and a remote client

In comparison to the previous type of communication, this type relates mostly to the application layer. It focuses on an application data exchange where data format has the form of messages. Protocol is responsible for the communication between HG and a remote client. Therefore, this communication puts emphasis on speed, reliability and security.

Our solution considers the usage of SIP right for this type of communication.

E. Communication between HG and clients

To support the communication, we think about three different types of abstract messages, SET, GET and NOTIFY that relate to distinct operations, which are performed by HG. The support of messages should be considered as the requirement on selecting correct protocol. The message exchange proceeds through interface of specific protocol in the process of implementation.

The GET message is used to send requests from the side of a remote client. HG reacts by sending requested data back in the next GET message. This way the client can get information about actual state of the home. To be able to send command and set a state, remote client uses the SET message. After receiving it, HG reacts by executing desired action. The initiator of communication in both previous cases is the remote client. To be able to initiate communication from the side of HG, the NOTIFY message is needed. Using this message HG informs remote clients about events, which just happened, e.g. detection of fire or breach of security. The communication protocol must allow simple and efficient implementation of all of these abstract messages and ensure their correct exchange. There are some other requirements. For example, video cameras are often used parts of the smart home. Therefore, there is also a requirement to support real-time transmission of video data from HG to the remote client. Other requirements come out from the security and innovations. There must be ensured that only authorized clients can communicate with HG and control its action. Similarly, we should be able to add new functionality without bigger problems, therefore the solution has to be technologically independent.

According to these requirements together with the openness, there are four communication protocol candidates – HyperText Transfer Protocol (HTTP) [10], eXtensible Messaging and Presence Protocol (XMPP) [11], MQ Telemetry Transport (MQTT) [12] and SIP.

The HTTP protocol is used for transfer of web content and it natively supports the GET method. It allows the implementation of GET and SET abstract messages. However, the protocol does not provide native and straightforward support of the NOTIFY message. One of possible ways how to implement this feature lies in uninterrupted sending of requests by client to HG. Once an event happens, HG will inform a client using following GET message. HTTP enables us to transfer video via streaming servers and Web Real-Time Communication (WebRTC) technology.

XMPP is a protocol based on XML objects. XMPP supports the implementation of all three abstract messages using appropriate sections of XML message – presence, message and iq (info-query). XMPP supports video transfer with the use of different extensions, as for example Jingle.

MQTT is an open communication protocol based on the client-server architecture. Since 2014 MQTT is one of OASIS standards. MQTT allows simple message transfer between various devices. Its implementation is very suitable for devices, which have limited performance, memory and power. These properties are typical for devices used within Internet of Things (IoT) or machine-to-machine (M2M) devices, for example those as various sensors, smart appliances and other embedded systems located in the home. Thanks to used communication model a MQTT server, which is called broker, is able to handle a great number of concurrently connected clients. The MQTT communication is based on the exchange of publish/subscribe messages and it uses a data-centric approach. A MQTT client does not send data directly to another client, but its data is forwarded only to the corresponding MQTT server. The server is afterwards responsible for "sharing" data to subsequent subscribed clients. The MQTT protocol supports enhanced security features like authentication, authorization and encryption through the implementation of the Transport Layer Security (TLS) protocol. As an advanced additional feature, the support of quality of service (QoS) can be included. MQTT usually runs on top of the TCP/IP architecture. However, taking into account actual progress in the field of IoT, there was created a new extension called MQTT for Sensor Networks (MQTT-SN) [13], which is designated for connectionless networks, as for example wireless sensor networks. MQTT-SN can be used on top of any kind of network, which provides lossless bidirectional connection with preservation of message order. Thanks to the network independence various transport protocols, like Zigbee known in the area of embedded devices, are available for MQTT-SN. Main goal and advantage of MQTT-SN is that it allows the deployment of MQTT in networks with low bandwidth connections and high link failures - wireless sensor networks. MQTT in its nature is only a simplified version of other here mentioned protocols, such as SIP and XMPP, however MQTT mainly concentrates on one specific function, the transfer of simple messages by minimizing their size and simplifying their transfer and processing needs. MQTT is appropriate and nowadays often used protocol alternative for a sensor data distribution within smart homes. On the other hand, its simplicity doesn't offer any method to meet some requirements mentioned above, such as video data transmission. To allow such services the collaboration with another protocol is required.

SIP has been proposed for a session management. Its role lies in creating, controlling and removing sessions between two and more end nodes. It describes each session with distinct parameters exchanged in specific messages. SIP is independent of used transport layer, provides unique abstract address for each device in the form of SIP URI and gives us means for simple implementation of all previously discussed abstract messages. SIP distributes individual functions among different logical entities as clients, registrar and proxy servers. The list of functions and their responsible entities includes registration, authentication and authorization of remote clients on the side of HG (SIP proxy), basic security, message exchange on the side of remote client (SIP Agent) and an interface between SIP and intern control plane of the smart home (SIP Gateway). Section **5** will provide a detailed description of proposed communication models, which incorporates SIP entities and their co-operation. An example of relation between SIP entities is shown in Fig. 1.

To ensure a secure connection, SIP supports several mechanisms. The simplest one uses a basic authentication based on a login name and a password. Others include the support of secure technologies like TLS, Message Digest 5 (MD5) and Internet Protocol Security (IPSec). All of these may be added to the communication model as means of enhanced security.

Considering all advantages of SIP over other three protocol candidates and our overall experience and knowledge gained by working with it, we decided to use the SIP protocol.

4. Architectural design

SIP allows several ways of implementation of communication between the remote client and HG. We propose three slightly different communication architectures. They describe relations between individual smart home components. They differ from each other in the placement of SIP server and in additional use of Virtual Private Network (VPN) entities.

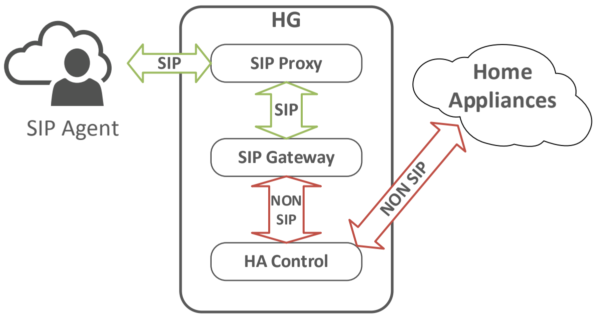

Fig. 1 Relations between SIP entities

F. *SIP-only public server-based architecture*

This architecture relies on the use of publicly placed SIP server with a public IP address assigned. Therefore, the server is globally accessible. The HG entity is a part of a private network located behind a NAT device inside the smart home. Both, the remote client (SIP Agent) and HG have to register to the public SIP server capabilities. After successful registration, communication between the client and HG goes via SIP server.

The advantage of this architecture is public SIP server accessibility where one individual SIP server may be used to control the communication of a group of smart homes. On the other hand, the charge for renting a public SIP server by some hosting company causes that the overall costs increase and requirement on the low price isn't more met. Other issue is the reliability of public SIP server, which represents a Single Point of Failure (SPOF) [14]. The risk of SPOF is usually carried out by implementing a cluster of public SIP servers where each is a backup for the others. This, of course, rises costs again. Figure 2 gives a better understanding of relations between given entities.

Considering advantages and disadvantages of this architecture, it could be effectively used in the case of multi-domain server deployment model (with more customers). However, for our view of the smart home it isn't suitable.

F. SIP-only private server-based architecture

As shown in the brief descriptive picture (Fig. 3), this architecture uses a private SIP server. The SIP server is implemented together with HG and both are placed in a private network of the smart home (i.e. behind customer's internet router with NAT). It means that each smart home has its own SIP server with assigned private IP address and with complicated remote reachability. The main issue lies in the translation of private address to the public one because the household gets from an ISP usually only one dynamically allocated public IP address. As a solution, which allows the access from a remote client to the SIP server placed in a private network, we can use Dynamic Domain Name Server (DDNS) together with Port-forwarding. DDNS informs the remote client about actual public IP of the home gateway. Port forwarding ensures that home gateway forwards each incoming SIP message (with destination port equal 5060) to the private address of SIP server.

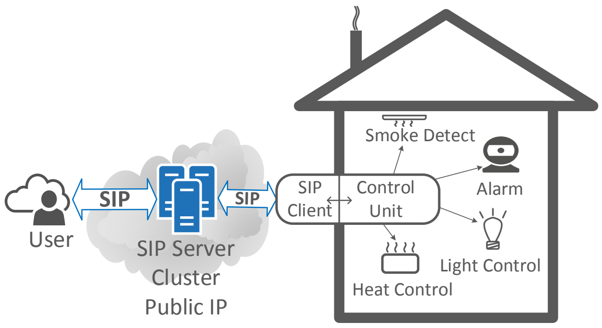

Fig. 2 SIP-only public server-based architecture

The existence of NAT presents especially for SIP a serious issue. The SIP protocol in its messages uses parameters of lower network layers like IP addresses and transport ports. There exist ways that address NAT in SIP like STUN and TURN. However, the overall functional implementation of aforementioned methods for NAT traversal is not straightforward.

H. SIP-VPN private server-based architecture

Looking at disadvantages of previous architectures, we can make an enhancement by adding additional features. For the interconnection of a client and HG a separate virtual private network (VPN) connection is created. To make it possible, VPN server runs beside SIP server. Both are placed in a private network. On the other side of connection, a VPN client is used as a part of remote client. Using VPN, the issue related to NAT is cleared, because both communication nodes (client and HG) are in the same network. In addition, VPN improves the security by using encrypted connection. The problem of learning actual public IP address of home gateway is also real in this architecture. To overcome the issue, DDNS and Port-forwarding techniques can be used again. Connections between individual parts of the architecture are in Fig. 4.

SPOF issue mentioned in the first proposed architecture occurs also in remaining two. In the case of second and third design, we have to back up HG by using backup power supply and a duplicated hardware device of implemented HG. Loss of internet connection opens a security problem, because without a connection the smart home cannot inform remote client about occurred events. Therefore, it is important to implement other ways of informing remote clients, which run simultaneously. For example, SMS message can be sent to the client's phone to inform or alert him.

According to requirements on the smart home, advantages and disadvantages of individual architectures, especially difficulty of implementation and security issues, the third one seems to be the right choice. Therefore, our implementation presented in the next section comes out from the SIP-VPN private server-based architecture.

I. Models of SIP message flows

As was already mentioned in section 3, there are several ways or models how to incorporate and use SIP to satisfy our needs. We again propose two communication models, which describe types and flows of individual SIP messages. Video streams aren't considered in following models because Real-time Transfer Protocol (RTP) is responsible for their transfer immediately after successful session establishment.

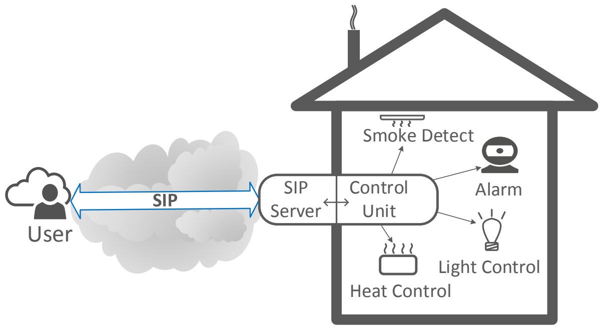

Fig. 3 SIP-only private server-based architecture

Communication models refer to various logical entities where each plays a particular role. SIP proxy manages SIP communication and includes client registration. SIP GW serves as an interface between a SIP proxy and a Home automation (HA) controller, which communicates directly with sensors. SIP GW consists of SIP server and intern SIP user agent (SIP

UA). The second SIP UA is used at remote client. From the SIP communication perspective, both act as end clients that communicate on a peer-to-peer basis. For clarity, HA controller is also put into visual figures of models and is responsible for management of sensors and control elements. Usually several logical entities are realized in one physical HW device. In our models SIP proxy, SIP GW and HA run as processes on one physical device.

Both models assume that both clients (SIP UA on remote client and SIP UA on SIP GW) are already registered at SIP proxy. Logical entities and their relations for each of following models are in Figs. 5 and 6.

1) Message-only model

The first model uses only one type of SIP request to implement all three abstract messages described in section 3 (GET, SET and NOTIFY). The SIP MESSAGE request is originally a part of Instant Messaging service (IM), which provides primitive chat functionality.

When a client wants to check actual value or a state of specific subsystem (GET operation), it sends request in the form of SIP MESSAGE (see Fig. 5). SIP MESSAGE includes requested parameters and instructions. The request is received at SIP GW.

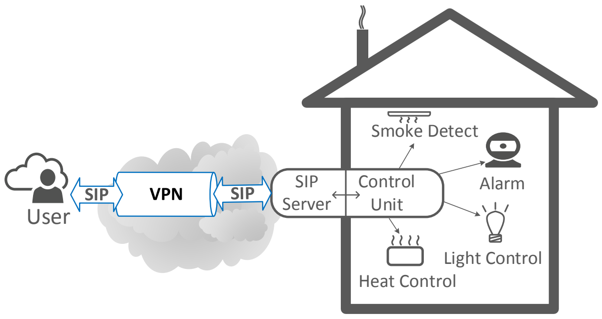

Fig. 4 SIP-VPN private server-based architecture

SIP GW extracts instructions from the message and sends them on the input of HA. The HA entity processes incoming instructions, reads data from desired sensor and as a response informs back SIP GW. In our solution we are using for this exchange means of inter-process communication. As a next step, SIP GW uses internal SIP UA to generate a SIP MESSAGE with desired value and sends it to the remote SIP UA.

In the case of SET abstract message almost the same flow is used. However, at HA there is distinct processing of the request where HA takes desired action (apply setting) instead of just reading values from sensors.

The issue emerges in the case of NOTIFY abstract message where we need to initiate SIP communication from HG. In this model we propose a simple solution based on regular sending of requests from remote client about actual state of monitored event.

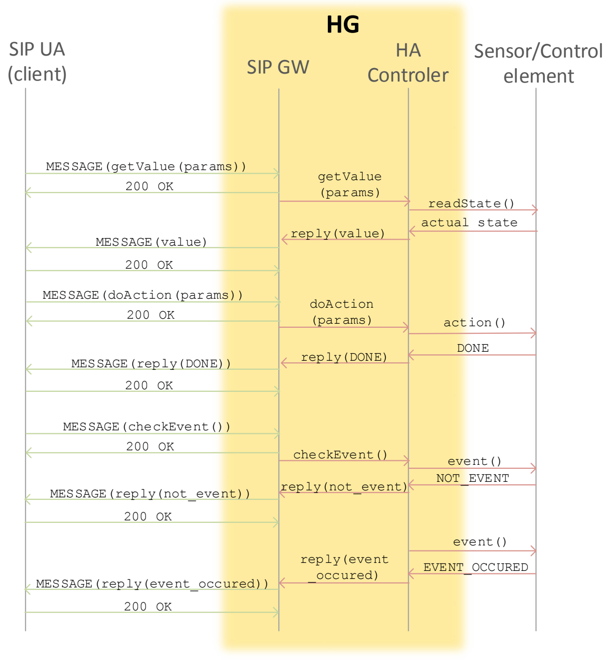

Fig. 5 Message-only model

The main advantage of this model lies in its simple implementation. On the other hand, proposed solution of NOTIFY message implementation is inefficient. Situation gets worse when more remote clients check occurrence of some event simultaneously. In that case, HG must handle each of them separately and that causes significant increase of HG load.

2) Message-presence model

The second model adds presence service with corresponding PUBLISH and NOTIFY SIP message types. This separates the implementation of NOTIFY and GET abstract messages. The SET message is implemented in the same way as it was in the previous model.

Used presence service requires deployment of additional SIP entities - watcher, presentity and presence server. Watcher is a subscriber that applies for the presence information about a specific subsystem or an event. Presentity is a monitored entity specified by its state. When the state is changed, information is sent to the presence server in the PUBLISH message. Presence server (PS) is responsible for keeping a list of subscribers. It maintains actual states of registered presentities. In the case of status changes, PS sends information about this change to all subscribers in the list. NOTIFY message serves this purpose. The body of both message types has unique format called the Presence Information Data Format (PIDF) and it has the form of XML document. The SIP presence extension is used for the implementation of NOTIFY and GET abstract messages.

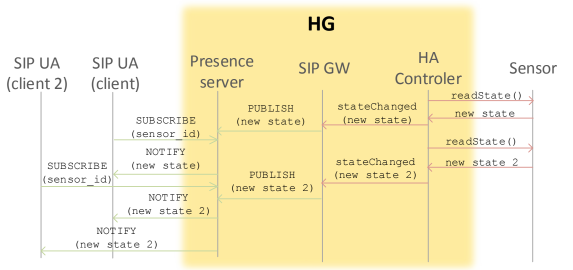

Fig. 6 Message-presence model

Monitoring of sensors is done by HA. Each change of a state is automatically reported to SIP GW (Fig. 6). SIP GW then creates a PUBLISH message, which includes this new state and sends it to the presence server. PS updates corresponding records. Subsequently PS sends the NOTIFY message to all registered subscribers. Therefore, the remote client has to subscribe to particular subsystem in order to get desired information about it.

This behavior of PS helps to solve the issue related to simultaneous checking of an event occurrence by more remote clients. Thanks to the deployment of SIP presence service, the whole responsibility is shifted from HA to the presence server. This solution is more efficient compared to the first solution and is considered as more preferable. Disadvantage of this model is its demanding implementation, which requires additional configuration of presence server and the support of presence service at both SIP UAs.

According to proposed models, their requirements and properties, we had to decide in our solution for the first model. The main reason comes from the lower difficulty of its implementation and the availability of suitable software components in comparison to the second model.

5. Our implementation

Summarizing our choices, as the proof concept we implemented the solution based on SIP-VPN private server-based architecture and message-only communication model. Description of the solution's components is provided in this chapter. The solution meets requirements specified in the introduction of the paper. We split the description into two parts - hardware and software. Each part contains specification of used tools and devices that correspond to individual smart home entities.

A. *Hardware components*

The list of HW components includes HG, sensors and control elements. HG is the main physical entity of the smart home, as mentioned earlier. HG is a HW device where all logical entities (client and servers described in section 4) will run as software processes. Therefore, HG must have sufficient performance to be able to perform all requested processes at the same time. In addition to that, HG must support sufficient amount of physical connectors and their types. Through them, we interconnect other parts like sensors and control elements. To be as close as possible to a real environment, we expect and support the use of following connectors. Ethernet port (RJ-45) for the internet connection, a set of General Purpose Input/Output (GPIO) pins used to manage sensors and control elements, 1-Wire bus interface for thermal sensors, an analog output for smoke sensor and a Universal Serial Bus (USB) port for video camera. One of key requirements requested on the HG is its power-consumption, which has to be low. Considering all requirements, a single board computer (SBC) was chosen as a hardware platform for the solution implementation. There are many SBCs on the market today. We compared three different models – Raspberry Pi 2 [15], Cubieboard 2 [16] and BeagleBone Black [17]. Based on the results of comparison shown in Table 1 we picked BeagleBone Black. In comparison with other two options, it meets all requirements and brings us additional advantages in the form of operating system stability and simple handling of peripherals.

For demonstration purposes as sensors, we implemented one sensor type for each of subsystems. As a thermal sensor, we used DS18B20 by Dallas Semiconductor. Sensor supports 1-Wire bus and assigned unique address. Thanks to the shared access of 1-Wire bus we are able to attach more sensors without a need for additional pins or connectors. DSN-FIR800 PIR sensor was chosen as an example of a motion sensor. As a smoke detector, we chose the MQ-4 sensor. It allows us to detect a potential fire. In addition, the smoke sensor allows to detect occurrence of a gas in the environment. For a water leakage detection, we used simple water sensor placed on the floor.

As control elements a set of relays controlled by GPIO pins is applied. Individual home appliances can be controlled via relay by changing a value of specific GPIO pin. We also demonstrate an electrical control of temperature. This is performed by electro thermic radiator head which is connected to a relay.

B. *Software components*

Logical entities of the smart home, including SIP and communication entities, are implemented as software components running in HG and in a remote client (smartphone). We expect entities: SIP server, SIP client, VPN server and a VPN client. As HG's operating system, we decided for the GNU/Linux. In the case of remote client Android OS is used. As a SIP server running on HG we selected the Kamailio SIP server. Kamailio provides all required functions including IM and Presence and its demands on the system resources are low. As a VPN server, which runs on HG, we used the OpenVPN server. OpenVPN server is free and stable IPSec solution. On a remote client we are running CSipSimple (a SIP client) and OpenVPN (VPN client) applications. Relatively complicated issue is an effective generating of SIP messages from HG. We solved this by using separate SIP clients running on HG. This option brings minimal additional delay and processor load without the need for the development of own clients. We picked the Linphone application as one of freely accessible and usually deployed SIP clients. Linphone almost supports all required functions including command line interface (CLI) management. CLI management allows simple control and is necessary for our solution. All used software tools are open-source as specified in requirements.

Comparison of three distinct SBCs Table 1

	BeagleBone Black	Cubieboard 2	Raspberry Pi 2
Processor	ARM-Cortex A8	ARM-Cortex A7	ARM-Cortex A7
Core	Single-core	Dual-core	Quad-core
CPU Frequency	1 GHz	1 GHz	900 MHz
Operational memory	512MB DDR3	1GB DDR3	1GB LPDDR2
Ethernet	✓	✓	✓
USB	1x	2x	4x
GPIO	65x	86x	40x
1-Wire	✓	✓	✓
Storage	2GB +SD card	4GB + SD card	SD card
Power Consumption	3W	4W	3W

Implementation of the HA controller requires special functionalities. Therefore, we designed and built our own software program written in C programming language. Based on ad-hoc solution, our controller is able to communicate and co-operate with SIP server as is required in the proposed **architecture**.

XI. HA controller design

Structure of the control program consists of one master and more slaves. Master is the main program responsible for initial start of slaves and their central management. Each slave is a separate unit implemented as a thread that provides particular functionality. We chose POSIX threads, which are available directly from Pthreads library. Threads require fewer resources than processes and offer simpler way of managing and sharing data between individual slaves. Communication from SIP server to HA is realized through simple file access, where after receiving specific SIP message SIP server extracts its body and stores it as text in the file. Next, these data are read by master of HA. For opposite direction HA uses a separate SIP client placed in HG to create and send a SIP message with desired information. Initial requests for creation and transmission of messages are in competence of individual slave programs. The flow of messages and mutual relations between parts described above is drawn in Fig. 7.

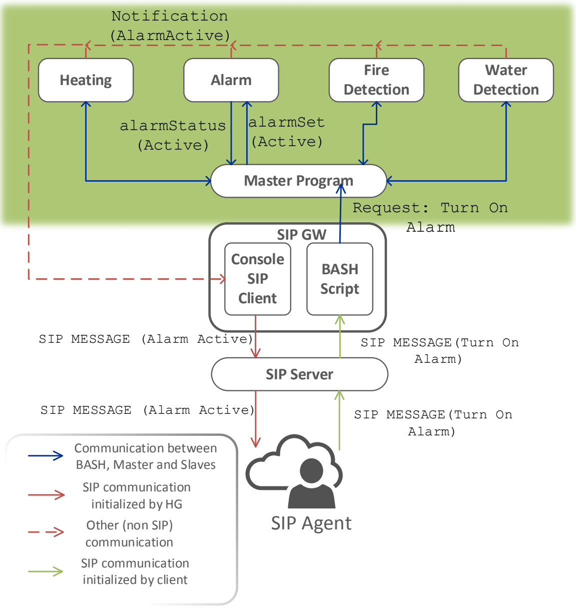
Notification (AlarmActive)
Heating
Alarm
Fire Detection
Water Detection
alarmStatus (Active)
alarmSet (Active)
Master Program
Request: Turn On Alarm
SIP GW
Console SIP Client
BASH Script
SIP MESSAGE (Alarm Active)
SIP MESSAGE(Turn On Alarm)
SIP Server
SIP MESSAGE (Alarm Active)
SIP MESSAGE(Turn On Alarm)
Communication between BASH, Master and Slaves
SIP communication initialized by HG
Other (non SIP) communication
SIP communication initialized by client
SIP Agent

Fig. 7 Communication model of HA controller parts

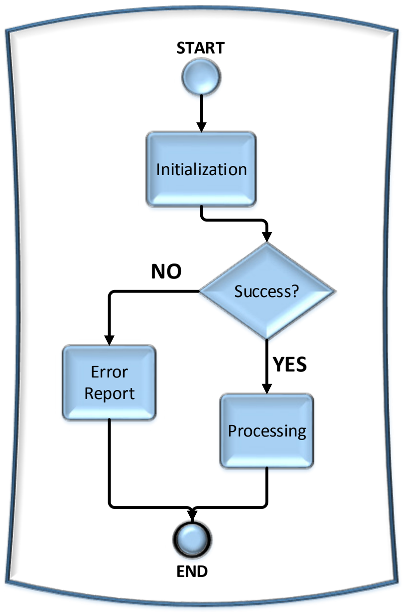

Fig. 8 Flow chart of common slave

Thanks to insertion of additional processing logic block into SIP server, every time it receives a SIP message destined for HA, it is able to identify action inside the message and store it in the text file. All slaves (heating, security, water and fire detection controllers) have very similar structure as shown in Fig. 8. They differ only in the way of processing particular function based on checking a state and reacting to its potential change. Slave periodically accesses data gathered by individual sensor to check its actual state. The activity has the form of access to the file. This abstraction is provided directly by operating system and its filesystem placed inside memory of our board PC. Subsequent reaction consists of creating appropriate message with measured value and sending it via SIP client interface to the desired destination. That is local SIP server or remote SIP agent.

The real power of this design comes from the distribution of functions into a set of simple programs running parallel in separate threads. It allows different subsystems to be controlled simultaneously and react to special events immediately. Although, our presented solution isn't optimal and requires additional improvements, it is good enough to show, that usage of the SIP protocol is so broad it can be used also in this manner.

6. Conclusion and future work

The paper describes the role of open protocols in the area of smart homes and is particularly dedicated to SIP. We addressed emerging issues related to this protocol. We analyzed recent papers focused on the problematic. Considering their results and principles, the paper introduces our basic smart home structure proposal with respect to present state of knowledge and research. The paper also provides an overview of solution's key components. We proposed three architectural models. Each of them uses slightly different approach to the realization of communication between the smart home and remote clients. Design, implementation and deployment of our solution were presented with emphasis on the realization of individual software and hardware components. We deployed a particular solution, which satisfies specified requirements, such as open-source approach, system autonomy, simplicity and minimal costs. Our testing solution costs about 120 €and provides functional system for basic management of the simple smart home.

There are some improvements of our solution, which should be realized as future works. For example, the realization and deployment of the second model mentioned in the paper. The intern SIP client, which is a part of SIP GW in actual solution, could be replaced with the use of functions directly from available SIP library inside of the HA control program.

References

[1] ROSENBERG, J. et al.: SIP: *Session Initiation Protocol*, IETF, RFC 3261, June 2002.

[2] SCHULZRINNE, H., WU, X., SIDIROGLOU, S., BERGER, S.: Ubiquitous Computing in Home Networks, *IEEE Communications Magazine*, pp.128-135, Nov. 2003.

[3] BERTRAN, B., CONSEL, C., KADIONIK, P., LAMER, B.: *A SIP-based Home Automation Platform: an Experimental Study*, Proc. of 13th Intern. Conference on Intelligence in Next Generation Networks, IEEE, Oct. 2009.

[4] ACKER, R., BRANDT, S., BUCHMANN, N., FUGMANN, T., MASSOTH, M.*: Ubiquitous Home Control based on SIP and Presence Service*, iiWAS '10 Proc. of the 12th Intern. Conference on Information Integration and Web-based Applications & Services, ACM, Oct. 2010.

[5] ROSENBERG, J.: *A Presence Event Package for the Session Initiation Protocol (SIP)*, IETF, RFC 3856, August 2004.

[6] KOVACIKOVA, T., SEGEC, P., BRUNCKO, M.: Standardization Paths for NGN IMS-based Architecture, *Communications - Scientific Letters of the University of Zilina*, vol. 10, No. 4, pp. 15-18, ISSN 1335-4205, 2008.

[7] SEGEC, P., KOVACIKOVA, T.: Implementation of IMS Testbeds using Open Source Platforms, *Communications - Scientific Letters of the University of Zilina*, vol. 14, No. 2, pp. 55-62, ISSN 1335-4205, 2012.

[8] DZINDZALIETA, R.: *SIP Protocol as a Communication Bus to Control Embedded Devices*, Local Proceedings and Materials of Doctoral Consortium of the Tenth International Baltic Conference on Databases and Information Systems, vol. 924, pp. 229-234, Vilnius, July 2012.

[9] GEBHARDT, J., MASSOTH, M., WEBER, S., WIENS, T.: *Ubiquitous Smart Home Control on a Raspberry Pi Embedded System*, The eighth Intern. Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 172-177, 2014.

[10] FIELDING, RESCHKE, J.: *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, IETF, RFC 7230, June 2014.

[11] SAINT-ANDRE, P.: *Extensible Messaging and Presence Protocol (XMPP): Core*, IETF, RFC 6120, March 2011.

[12] OASIS Standards: MQTT Version 3.1.1, 2014.

[13] STANDFORD-CLARK, A., TRUONG, H. L.: MQTT For Sensor Networks (MQTT-SN), Protocol Specification, Version 1.2, 2013.

[14] REZAC, F., et al.: Bruteforce Attacks Blocking Solution on Embedded SIP Communication Server, *Communications - Scientific Letters of the University of Zilina*, vol. 15, No. 2A, pp. 180-184, ISSN 1335-4205, 2013.

[15] UPTON, E.: https://www.raspberrypi.org, 2015.

[16] CUBIE, T.: http://cubieboard.org, 2013.

[17] BeagleBoard Foundation, http://beagleboard.org/black, 2013.