Autonomic Smart Home Operations Management using CWMP: A Task-centric View

Chun-Feng Liao, Shih-Ting Huang, and Yi-Ching Wang

Department of Computer Science
National Chengchi University, Taipei, Taiwan
{cfliao,101703006,103753028@nccu}@nccu.edu.tw

Abstract. Despite the well-development of smart living space research field, Smart Home is still more like a luxury product than a daily necessity for most families. Operations management issues are essential for a new technology to be accepted by the mass consumer market. However, only few attempts have been made toward this direction. In this paper, we present the design and implementation of a CWMP-based platform that supports autonomic operations management. The experiment results show that the proposed approach is stable and is able to drive the operations tasks smoothly. We also demonstrate the feasibility of the platform by realizing two application scenarios supported by the prototype of the proposed approach.

Keywords: CWMP, TR-069, Operations Management, Smart Home

1 Introduction

The concept of Smart Home was envisioned twenty years ago [1]. Although the essential technology for constructing smart living spaces has also been an object of study for more than two decades [2] and the costs of embedded computers, sensors, and home appliances are much lower than before in the last few years, Smart Home is still more like a luxury product than a daily necessity for most families. As pointed out in recent researches, central to this issue is the problem of autonomic operations management, namely, the ability of a Smart Home system to be self-deployable[3], self-diagnosable[4, 5], and self-configurable [6, 7]. The above-mentioned self-* properties of systems have been proposed by the researchers in the field of "autonomic computing" [8].

Despite the importance of operations management in Smart Home, only few attempts have been made toward this direction. Rachidi and Karmouch's work [9] is a pioneer study in this issue. Based on the MAPE-K (Monitor, Analyze, Plan, and Execute using Knowledge) model of autonomic computing, they present an approach that facilitates self-configuration for home gateway based on CWMP (CPE WAN Management Protocol, where CPE abbreviates Consumer Premises Equipments) [10]. CWMP, propose by Broadband Forum and also know as TR-069, is a SOAP-based[11] application layer protocol for remote management

and configuration of CPEs, where CPEs are usually home gateways in real-world applications. As CWMP has been implemented on more than 250 million devices world-wide [12], it is apparently a good basis for designing the Smart Home operations managing mechanisms.

Figure 1 is an UML deployment diagram that illustrates a typical application architecture of autonomic operations management using CWMP. The service provider's server hosts an MAPE-K Manager module to proactively analyze, plan, and determine the strategies of management. After a decision is made, several commands are executed by an ACS (Auto-Configuration Server). In a typical scenario, an ACS is responsible for performing administrative operations of home gateways and a CPE is hosted by a home gateway. Generally, a home gateway is usually equipped with one or more protocol gateway modules to HAN (Home Area Network). For instance, to enforce the management strategy remotely in an UPnP-based home network, an UPnP Control Point module has to be implemented and deployed in the home gateway. Figure 1 also reveals that CWMP's scope is limited to communication among an ACS and CPEs.

In fact, the CWMP specification only defines signatures of remote procedures (listed in Table 1) and their SOAP representations, figuring out how to compose these remote procedures so that certain operations management tasks can be carried out is a burden of developers. In other words, CWMP specification [10] does not define how to orchestrate these methods to perform operations management tasks.

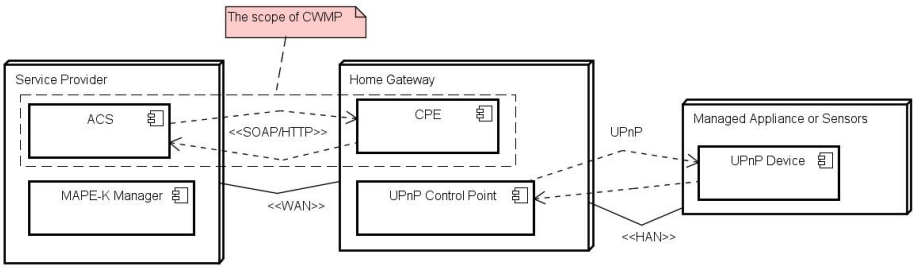

Fig. 1: A typical autonomic operations management architecture using CWMP

Inspired by Rachidi and Karmouch's work [9] on CWMP-based self-configuration, this paper aims to design and to implement other two essential self-* mechanisms, namely self-deployment and self-diagnosis, that supports MAPE-K-based autonomic operations management using CWMP. To relief the burden of developers, this work takes a task-centric approach. Specifically, we design the system by first identifying the core tasks that is critical to a Smart Home's daily operations management. Then, we examined and identified the call sequence of CPE/ACS operations that collaboratively accomplish these tasks. To evaluate the proposed call sequences, this paper demonstrates the feasibility of the proposed approaches

Table 1: Core Methods specified in CWMP

Subject	Method Name	Subject	Method Name
CPE	GetRPCMethods	ACS	GetRPCMethods
CPE	SetParameterValues	ACS	Inform
CPE	GetParameterValues	ACS	TransferComplete
CPE	GetParameterNames	ACS	AutonomousTransferComplete
CPE	SetParameterAttributes	CPE	GetParameterAttributes
CPE	AddObject	CPE	DeleteObject
CPE	Reboot	CPE	Download

by implementing a prototype and by conducting experiments on the prototype. We hope that our work is helpful for the developers of infrastructure for Smart Home service providers when developing operations management tasks.

2 Related Work

The advent of Smart Home brings about the issues of operations management. Operations management means the configuration, deployment, upgrade, and monitoring of devices or systems in Smart Home. Obviously, the best way to address this issue would be to reuse an existing mature standards or technologies. In network equipment industry, several standards have been proposed for operations management such as NETCONF[13], MUWS (Management for Using Web Service)[14], WS-Management[15] and CWMP[10]. As mentioned, CWMP has been widely deployed in home gateway, and therefore are considered the most competitive among the standards[9].

As a result, several works have been done for developing management functions based on CWMP. Nikolaidis et al.[6] proposed an MIB-based (Management Information Base) framework and a graphical development environment for control devices in a home network. Other works focused on providing the tools to assist service engineers to diagnose hazards in Smart Home system [16, 17]. There are relatively fewer studies aim to realize the operations management in a Smart Home. Rachidi and Karmouch [9] are one of the pioneering works toward this direction. Based on CWMP, they presented an approach for realizing self-configuration using MAPE-K model of autonomic computing. This work differs from Rachidi and Karmouch's work in that we take a task-centric approach and focus on different operations management issues, namely self-deployment and self-diagnosis.

3 Design

In this section, we shall present the approaches for supporting self-deployment and self-diagnosis using CWMP. Before turning to the details of design, the essential tasks of deployment and diagnosis must be clarified first.

- **Deployment tasks:** without self-deployment, a service engineer is required to perform on-site setup and configuration tasks for newly installed system. Moreover, when users want to buy a new service or to upgrade an existing service, a service engineer also must be present. Thus, the deployment tasks are labor intensive. In this work we focus on the following deployment tasks: 1) setup a newly installed system; 2)download and install a new software module, which is called a Deployment Unit (DU), in CWMP; 3) update an existing DU.
- **Diagnosis tasks:** It is widely recognized that robustness is a paramount concern of Smart Home users [18, 19], since most of the domestic technologies are expected to work 24-7. The occupants of Smart Homes are usually non-technical users, so that Smart Home is in lack of professional system administrator. Since the consumers would be unable to pinpoint the source of failures [20], the Smart Home system must be highly reliable and be able to detect and to recover from failures autonomously.

Having considered the essential tasks of self-deployment and self-diagnosis, let us now turn to detailed mechanisms for accomplishing these tasks, which are presented in the following sub-sections.

3.1 Supporting Self-Deployment Tasks

Ease of installation is a key factor that determines if a new technology can be accepted by mass consumer market. As mentioned, installation tasks including setting up a new system, a new deployment unit and upgrade an existing deployment. These tasks can be automated by a defined sequence of CWMP interactions between CPE and ACS, as shown below.

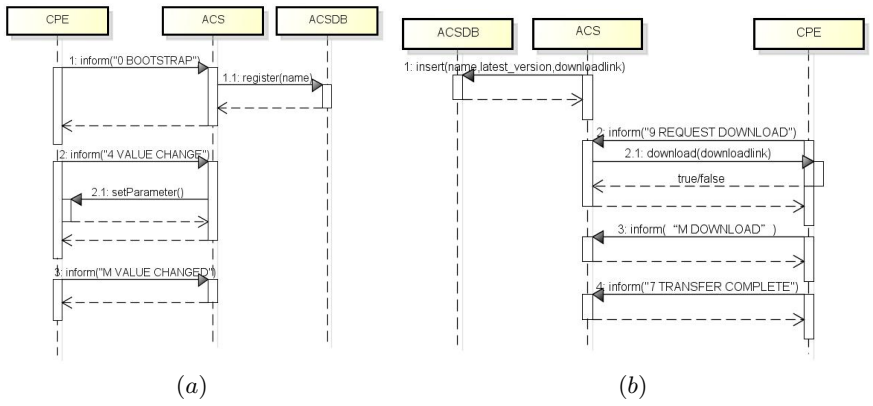

Fig. 2: Supporting self-deployment tasks: (a) setup a newly installed system (b) install a new deployment unit

Setup a newly installed system: As depicted in Fig.2a, when a CPE is installed and boots for the first time, it establishes a connection to ACS using the factory-default IP address. CPE then sends an **Inform 0 BOOTSTRAPE** to ACS, which is an event notification indicates that the CPE boots for the first time. Meanwhile, CPE also register its identity in ACS's database (ACSDB) for further management. After that, CPE sends an **Inform 4 VALUE CHANGE** to ACS, and then ACS calls CPE's **setParameter** method to configure the CPE. Finally, CPE sends **Inform M VALUE CHANGED**, which means auto-configuration of a newly installed system has been finished.

Install a new deployment unit (DU): In this task, we assume that the user buys a new DU and users CPE is responsible for downloading and installing the DU. After the DU is ready for deployment, ACS inserts the new DUs name, latest version, and download link into a table in ACSDB. Then, CPE sends **inform 9 REQUEST DOWNLOAD** to ACS and ACS calls **download** method of CPE and pass the download link and access token as arguments. After the download process is complete, CPE sends **inform 7 TRANSFER COMPLETE** to ACS. Finally, ACS updates the current DU version number in ACSDB. The overall process is indicated in Fig.2b.

Check and upgrade a deployment unit: After a DU is installed, CPE periodically sends **inform 2 PERIODIC** to ACS to check the version of DU. If there is a new version available, CPE sends **inform 9 REQUEST DOWNLOAD** to ACS, and then ACS calls CPE **download** and pass the download link and access token as arguments. After finishing the download, CPE sends **inform 7 TRANSFER COMPLETE** to ACS. Again, the last step is to update the current DU version number in ACSDB. The overall process is indicated in Fig.3.

3.2 Supporting Self-Diagnosis Tasks

As the consumers are usually unable to pinpoint the source of failures, it is desirable that a Smart Home system being able to detect and to recover from failures autonomously. In the following, we present our design of realizing self-diagnosis tasks for CPE and DU using CWMP.

Diagnosing CPE: CWMP defines a set of CPE methods to monitor the status of CPE. CPE can periodically send **inform 2 PERIODIC** to ACS to indicate the "liveness" of itself. This mechanism is usually called "heartbeat". Once ACS does not receive the heartbeat from CPE, it first tries to reboot CPE by calling the **Reboot** method. If CPE does not respond, it assumes that there is something wrong with CPE and then contacts the service provider automatically. The overall procedure is shown in Fig.4a.

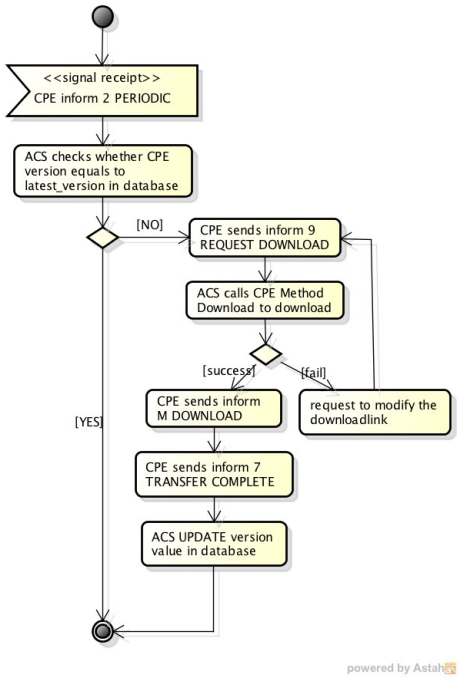

Fig. 3: Supporting self-diagnosing tasks: Check and upgrade a deployment unit

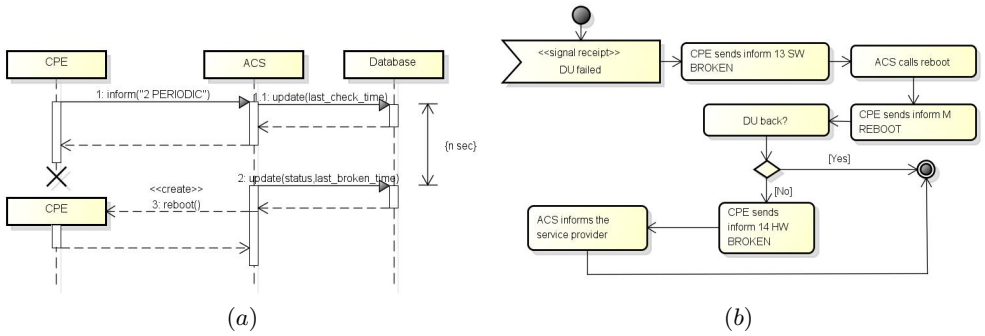

Fig. 4: Supporting self-diagnosing tasks: (a) diagnose CPE (b) diagnose and recover DU

Diagnosing deployment units: Figure 4b depicts the process of diagnosing and recovering of DU. There are typically several DUs being installed on a CPE. As the original CWMP does not define event types that indicate software of hardware failures, we defined two CWMP inform extension using the **inform X** prefix, namely **inform X HW BROKEN** and **inform X SW BROKEN**. When CPE finds a DU fails, the CPE sends **inform X SW BROKEN** to ACS, and then ACS calls the CPE **Reboot** method (try to recover DU). After the CPE is rebooted, it sends **inform M REBOOT** to ACS. If the problem still exists, CPE sends **inform X HW BROKEN** to ACS and then ACS contacts the service provider automatically.

4 Implementation

This section discuss the implementation issues of our design. Currently, we implement a working prototype using JAX-WS (Java API for XML Web Services) [21]. JAX-WS is a Java API for creating web services. It is mainly used to build web services and corresponding clients that communicate using XML-based remote procedure calls, which implemented based on SOAP[11]. The detailed implementation mechanisms for accomplishing the self-deployment and self-diagnosis tasks are presented in the following sub-sections.

4.1 Self-Deployment Tasks

As mentioned, deployment tasks are 1) setup a newly installed system, 2) install a new deployment unit, and 3) check and upgrade a deployment unit. The detailed steps of implementing setup a newly installed system task is listed below.

```
Step1: CPE boots
Step2: CPE sends "inform 0 BOOTSTRAP"
Step3: ACS process inform
Step4: CPE sends "inform 4 VALUE CHANGE"
Step5: ACS calls the "setParameterValue" method
Step6: CPE sends "inform M VALUE CHANGED"
```

To implement the setup a newly installed system task, the CPE Inform and ACS setPrarmeterValue methods are being used. According to the CWMP specification, argument type of the CPE Inform method is *EventStruct*, which is used to carry required information for the notification. When CPE boots, CPE sends **inform** to ACS to notify CPE has booted and then ACS uses **setParameterValue** method to set values which need to be initialized. To implement the install a new DU and upgrade DU task, we use the JDK **TimerTask** class to make CPE send inform to ACS periodically. Then, getParameterValue is called to get the version number of a DU. Upon a new install or an upgrade is required, the **downlaod** method of CPE is called. The following procedure indicates steps of DU download and upgrade.

```
Step1: CPE regularly]e.g. every n seconds^sends "inform 2 PERIODIC"
Step2: ACS calls the CPE "getParameterValue" to get CPE version number
Step3: ACS compares the version number with latest_version value in ACSDB
Step3: If version number < latest_version, CPE sends "inform 9 REQUEST DOWNLOAD"
Step4: ACS calls the CPE "download"
Step5: CPE sends "inform M download"
Step6: CPE sends "inform 7 TRANSFER COMPLETE"
Step7: Back to Step1
```

4.2 Self-Diagnosis Tasks

As mentioned, deployment tasks are 1) diagnose CPE and 2) diagnose DU. Let us first take a look at the detailed steps of diagnosing CPE.

```
Step1: CPE regularly (e.g. n seconds) sends "inform 2 PERIODIC"
Step2: ACS receives heartbeats from CPE
Step3: ACS does not receive the inform from CPE for a period of time
Step4: ACS calls the CPE "reboot" method
Step5: CPE sends inform "M reboot"
Step6: ACS does not receive the inform from CPE for a period of time
Step7: ACS notifies service provider that the CPE has hardware failure
```

Similar to the check and upgrade a deployment unit task, CPE regularly sends heartbeat to ACS as heartbeat messages. When ACS does not receive heartbeat from CPE, the ACS first tries to reboot the CPE and if it still not receive the heartbeat form CPE, the ACS notifies service provider that the CPE has hardware failure. The diagnose DU task starts when the CPE finds that DU fails. Generally, this is reasonable as CPE and DU located in the same host. CPE then sends inform to ACS to notify that DU is down. Again, ACS first tries to restart the failed DU and then if it is still not recovered then ACS notifies service provider that the CPE has hardware failure. The overall process is listed below.

```
Step1: CPE finds that the DU fails
Step2: CPE sends "inform 13 SW BROKEN"
Step3: ACS calls the CPE "reboot" method
Step4: CPE sends "inform M REBOOT"
Step5: If DU still not available
Step6: CPE sends "inform 14 HW BROKEN"
Step7: ACS notifies service provider that the DU has unrecoverable failure
```

5 Evaluation

To evaluate the proposed approach, we first build two application scenarios to verify the feasibility. Then, we conducted experiments to test the performance of performing deployment and diagnosis tasks using the proposed approach.

5.1 Feasibility

We constructed application scenarios based on the prototype detailed in Section 4 to show the feasibility. In these scenarios, it is assumed that the user buys the TV media service, and that the service contains two DUs: TV controller and a media server. The CPE prototype is implemented and deployed on a Raspberry Pi Model B+ model and the ACS prototype is deployed on a normal PC.

Deploying the TV media service First, CPE downloads the media server DU. Then, the DU is downloaded and installed on CPE. From CWMP's perspective, the objective is to install a new DU so that the operations include to check the CPE version number and sends an **inform 9 REQUEST DOWNLOAD**. After the ACS is ready, it calls **download** of CPE. Finally, CPE replies **inform 7 TRANSFER COMPLETE** to finish this transaction. After that, the user can enjoy the TV media service, which is provided by connecting TV to the media server.

Diagnosing the TV media service As the objective is to diagnose the DU and recover the DU from failures if necessary. In this scenario, the media server DU does not respond **inform 2 PERIODIC** for a period of time. CPE detects the problem and informs ACS. Then, ACS first reboots the broken DU by calling **reboot** method of CPE, then CPE responds **inform M reboot**. Unfortunately, ACS finds that the DU still not responding. Thus, CPE informs ACS about this failure via email service and the service provider sends a service engineer to the user's home to repair the service.

5.2 Performance

This section reports the experiment results that are performed to evaluate the required time to download a DU file with various sizes and to study how the quantity of DUs impacts the performance of diagnosing.

Performance of Deployment Tasks This experiment runs deployment tasks and measure the total time required starting from CPE downloading to the completion of installation. We respectively test the file sizes ranges from 10KB to 10MB. Figure 5 shows the experiment results. The experiment is performed repeatedly using 10, 20, 40 and 60 CPEs. The turnaround time appears to increase linearly as the DU size increases. However, there is a steeply change after the DU size is larger than 1MB. However, as the general size of DUs typically ranges from 100K to 200K, which, according to the results, can be downloaded within 10 seconds. Also,the turnaround time also increases slightly as the number of CPEs increases. The increasing trend of turnaround time is consistent among experiments with different sizes of CPEs.

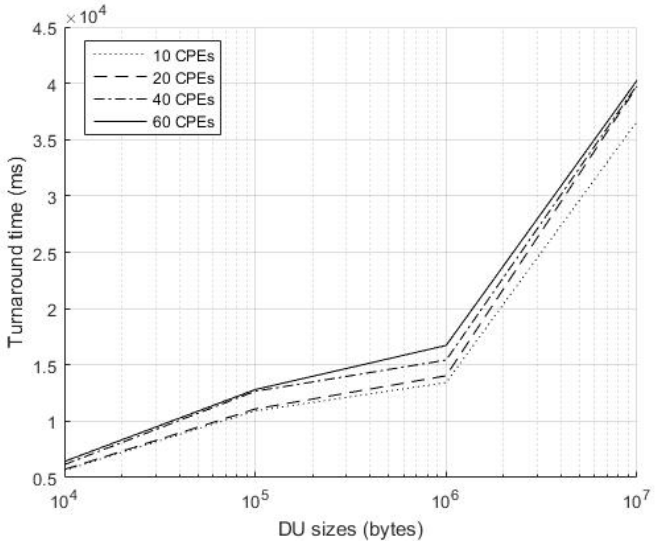

Fig. 5: Performance evaluation of DU deployment

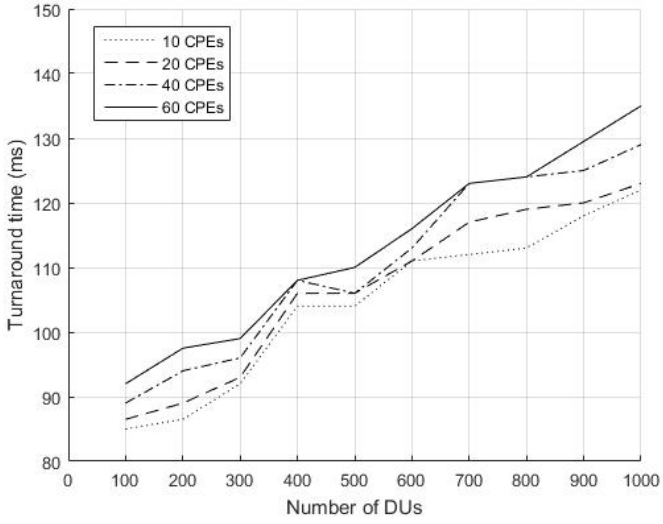

Fig. 6: Performance evaluation of DU diagnosis

Performance of Diagnosis Tasks In this experiment, we test the impact to performance of DU monitoring when number of DU increases. The test is performed when the number of DUs is 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000. The experiment is performed repeatedly using 10, 20, 40 and 60 CPEs. The results are depicted in Fig.6, which shows that the increase of DU number leads to the linear growth of monitor time. Based on this result, the monitor can be finished within 10 seconds with a reasonable number of DUs (100-200). Also,the turnaround time also increases slightly as the number of CPEs increases. Again, the increasing trend of turnaround time is also consistent among experiments with different sizes of CPEs.

6 Conclusion

This paper present approaches for supporting MAPE-K control loop between the service provider and the gateway of a Smart Home. Specifically, these approaches enables self-deployment and self-diagnosis of a Smart Home system based on CWMP. To relief the burden of developers, we take a task-centric approach, that is, we first identify the essential tasks for operations management in Smart Home and then define the interactions among entities using CWMP methods. The results show that both deployment and diagnosis have good performance when the DU size and DU number is reasonable. This paper present our fist step toward the vision of autonomic operations management. In the future, we are going to investigate how to transparently integrate the smart home network such as UPnP with the CWMP so that the devices in the home area network become also manageable.

Acknowledgements. This work is sponsored by Ministry of Science and Technology, Taiwan, under grant 104-2221-E-004-001 and 104-2815-C-004-008VE.

References

1. Gates, B., Myhrvold, N., Rinearson, P., Domonkos, D.: The road ahead. (1995)
2. Caceres, R., Friday, A.: Ubicomp systems at 20: Progress, opportunities, and challenges. IEEE Pervasive Computing (1) (2011) 14–21
3. Hnat, T.W., Srinivasan, V., Lu, J., Sookoor, T.I., Dawson, R., Stankovic, J., Whitehouse, K.: The hitchhiker's guide to successful residential sensing deployments. In: Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, ACM (2011) 232–245
4. Mennicken, S., Vermeulen, J., Huang, E.M.: From today's augmented houses to tomorrow's smart homes: new directions for home automation research. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM (2014) 105–115
5. Cetkovic, M., Nemet, N., Samardzic, T., Teslic, N.: Auto-configuration server architecture with device cloud cache. In: Consumer Electronics Berlin (ICCE-Berlin), 2014 IEEE Fourth International Conference on, IEEE (2014) 296–298

6. Nikolaidis, A.E., Papastefanos, S.S., Stassinopoulos, G., Drakos, M.P.K., Doumenis, G., et al.: Automating remote configuration mechanisms for home devices. Consumer Electronics, IEEE Transactions on **52**(2) (2006) 407–413
7. Feminella, J., Pisharoty, D., Whitehouse, K.: Piloteur: a lightweight platform for pilot studies of smart homes. In: Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, ACM (2014) 110–119
8. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1) (2003) 41–50
9. Rachidi, H., Karmouch, A.: A framework for self-configuring devices using tr-069. In: Multimedia Computing and Systems (ICMCS), 2011 International Conference on, IEEE (2011) 1–6
10. Bernstein, J., Spets, T.: Cpe wan management protocol. In: absorption in the earth's atmosphere, DSL Forum, Tech. Rep. TR-069. (2004)
11. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S., Winer, D.: Simple object access protocol (soap) 1.1 (2000)
12. Zapata, J., Fernández-Luque, F.J., Ruiz, R.: Wireless sensor network for ambient assisted living. Wireless Sensor Networks: Application-Centric Design. InTech (2010) 127–146
13. Enns, R., Bjorklund, M., Schoenwaelder, J.: Netconf configuration protocol. Network (2011)
14. Murray, B., Wilson, K., Ellison, M.: Web services distributed management: Muws primer. OASIS WSDM Committee Draft (2006)
15. Arora, A., Cohen, J., Davis, J., Golovinsky, E., He, J., Hines, D., McCollum, R., Milenkovic, M., Montgomery, P., Schlimmer, J., et al.: Web services for management (ws-management). Distributed Management Task Force (DMTF) (2004)
16. Bjelica, M.Z., Golan, G., Radovanovic, S., Papp, I., Velikic, G.: Adaptive device cloud for internet of things applications. In: Consumer Electronics-China, 2014 IEEE International Conference on, IEEE (2014) 1–3
17. Nemet, N., Radovanovic, S., Cetkovic, M., Ikonic, N., Bjelica, M.Z.: User self-help module for a device management cloud based on the tr-069 protocol. In: Consumer Electronics Berlin (ICCE-Berlin), 2014 IEEE Fourth International Conference on, IEEE (2014) 199–201
18. Edwards, W.K., Grinter, R.E.: At home with ubiquitous computing: Seven challenges. In: Proc. 3rd International Conference on Ubiquitous Computing (UbiComp'01). (2001) 256–272
19. Grimm, R., Davis, J., Hendrickson, B., Lemar, E., MacBeth, A., Swanson, S., Anderson, T., Bershad, B., Borriello, G., Gribble, S., Wetherall, D.: Systems directions for pervasive computing. In: Proc. 8th Workshop on Hot Topics in Operating Systems. (2001)
20. Dixit, S., Prasad, R. In: Home Networking Challenges. Wiley-Inderscience (2008)
21. Kohlert, D., Gupta, A.: The java api for xml-based web services (jax-ws) 2.1 (2007)