CloudThings: a Common Architecture for Integrating the Internet of Things with Cloud Computing

Jiehan Zhou, Teemu Leppänen, Erkki Harjula,
Mika Ylianttila, Timo Ojala
University of Oulu
Oulu, Finland
{firstname.lastname}@ee.oulu.fi

Chen Yu, Hai Jin, Laurence Tianruo Yang
Huazhong University of Science and Technology
Wuhan, China
yuchen@hust.edu.cn,
jinhust@hust.edu.cn,ltyang@ieee.org

***Abastract*—The Internet of Things presents the user with a novel means of communicating with the Web world through ubiquitous object-enabled networks. Cloud Computing enables a convenient, on demand and scalable network access to a shared pool of configurable computing resources. This paper mainly focuses on a common approach to integrate the Internet of Things (IoT) and Cloud Computing under the name of CloudThings architecture. We review the state of the art for integrating Cloud Computing and the Internet of Things. We examine an IoT-enabled smart home scenario to analyze the IoT application requirements. We also propose the CloudThings architecture, a Cloud-based Internet of Things platform which accommodates CloudThings IaaS, PaaS, and SaaS for accelerating IoT application, development, and management. Moreover, we present our progress in developing the CloudThings architecture, followed by a conclusion.**

Keywords – Cloud computing; Internet of Things; 6LowPAN; CoAP; RESTful Web services

I. INTRODUCTION

In 1999, Kevin Ashton [1] predicted that future computing would depend on more data captured by computer-enabled objects or things, rather than on data originated by people. Thus he brought the idea of the Internet of Things, in which objects are identified and are able to perceive or understand surrounding data. Objects are connected and are able to interact with servers over the Internet. Objects are also able to make queries and change their states and or their information programs. Ultimately, we would be able to track and count everything in the world, and greatly reduce waste, loss, and cost [2]. Nicholas Negroponte (head of the Media Lab at MIT) also claimed that the Internet of Things is about embedding intelligence so that things become smarter and do more than they did before. This paper takes a further step, and refers to the Internet of Things as all IPv6 addressable things operated with Web-based services – whether they are physical or virtual things. We use the term of Things to refer to all computer-embedded objects which operate on the Internet.

The Internet of Things presents widened opportunities and applications, including smart grids to improve efficiency and reliability of power supplies; intelligent transportation to optimize traffic management and reduce traffic accidents, clogged routes, and carbon dioxide emissions; environmental monitoring to oversee drinking water sources and urban atmospheres or supervise the transmission of dangerous wastes, or e-health to accelerate and coordinate management of medical information, hospital wards, patient care, and drug provision. However, there are many challenges facing Things-related application development, such as end user scalability, data storage, heterogeneous resource-constrained Things, variable geospatial deployment, or energy efficiency [3][4].

Cloud computing [5] creates a new way of designing, developing, testing, deploying, running and maintaining applications on the Internet. Traditionally, the application developer needs to take care of running operating systems, networks, load balancing, routers, firewalls, and storage, while integrating these things and allowing them to interact with the system. The developer also needs to take into account of scalability, or how the application could scale many geographically distributed users. Cloud computing applies a utility model to produce and consume computing resources, in which the Cloud abstracts all types of computing resources, including storage, as services (i.e. Cloud services). The Cloud user (either application developer or application consumer) can access the Cloud services over the Internet, and the Cloud users pay only for time and services they need. The Cloud can also scale to support large numbers of service requests. Ultimately, Cloud computing takes care of the micro-lifecycle management of applications, and allows application managers to focus on application development and monitoring. The Cloud computing platform is designed to consist of a variety of services for developing, testing, running, deploying, and maintaining applications on the Cloud. Examples of Cloud computing platforms are The Amazon Web Services [6], Google App Engine [7], and Microsoft's Windows Azure platform [8].

The Internet of Things and Cloud computing are both emerging technologies and have their own features. Things are

linked to their virtual representations on the Internet and are accessible via the Internet (i.e. Things as services, e.g. service-oriented pervasive computing) [9]. Cloud computing also applies the utility model, and enables end-users to accommodate and consume services in an efficient and pay-per-use way. In Mammoth project,[1] we raise the following question: Can we integrate Cloud computing into the Internet of Things, and accelerate Things application, development, and management?

To answer this question, our paper presents the state of the art for integrating Cloud computing and the Internet of Things. By examining a prototyped scenario, we analyze the IoT application requirements and present CloudThings architecture – a Cloud-based Internet of Things platform. The remainder of the paper is organized as follows: Section 2 reviews research on integrating Cloud computing into the Internet of Things. Section 3 examines cases of application in a typical IoT scenario. Section 4 studies Things application system requirements. Section 5 presents CloudThings architecture, a Cloud-based Internet of Things platform. Section 6 presents implemented prototypes towards CloudThings architecture, and then offers conclusions.

II. THE CLOUD-BASED INTERNET OF THINGS: A REVIEW

In Cloud computing, most of the computing resources exist on the Internet on servers, as opposed to client machines such as laptops or personal computers. Cloud computing is commonly associated with Information Technology (IT) services, but can theoretically be extended to embedded software programming [10]. Integrating Cloud computing with Wireless Sensor Networks (WSNs) brings the concept of Cloud-based embedded system programming. The Cloud-based integrated programming environment has a common benefit, namely that the local administrators and users don't need to spend time with large client and server machine installations, setups, or software updates. With Cloud-based tools, the user can program from anywhere that has an Internet connection. In the proposed Cloud-based model [10], the Cloud connects devices such as PCs, smart phones, embedded development platforms, or host machines to Cloud-based programming tools. These tools can include sales databases, or Integrated Development Environments (IDE), and can compile resources hosted by Cloud computing platforms such as Amazon.com, Microsoft, Google, and Yahoo. The Cloud-based model [10] entails that the Web-based tools are operating systems, and are client machine-agnostic. A further advantage of the Cloud model is the flexibility of implementation.

In RFID application development, Dominique et al. [11] pointed out that the deployment of RFID applications often remains complex and costly, since they involve the tedious deployment and management of large and heterogeneous distributed systems. Consequently, they are only suitable for large organizations; rather than the limited resources of small business applications. To address this problem, Dominique et al. discussed a Cloud computing solution integrating virtualization technologies and the architecture of the Web and its services. They applied the Amazon Web Service platform and Elastic Compute Cloud (EC2) services. The EC2 service allows the creation and management of virtual machines (Amazon Machine Images, or AMIs) that can then be deployed on demand onto a pool of machines that are hosted, managed, and configured by Amazon. The benefit of this approach is that the server-side hardware maintenance is delegated to the Cloud provider. Also it offers better scaling capabilities, as the company using the Cloud AMI, can deploy additional and more powerful instances according to the amount of requests.

The Cosm [12] (formerly Pachube) service for the Internet of Things provides data management infrastructure for sensors, devices, and environments. It is an on-line database service that allows developers to connect sensor data, e.g. energy and environment data, from objects to the Web, and to build their own applications based on that data. The Cosm manages millions of data points per day from thousands of individuals or organizations around the world. The Cosm allows people to embed real time graphs in websites. It analyzes and processes historical data pulled from any public data source Cosm feeds and sends real time alterations from any data stream to control scripts, devices, or their environments.

Nimbits [13] is an open source data logging Cloud server built on Cloud computing architecture that provides connectivity between the Internet of Things using data points. Users can use Nimbits to record and share sensor data on the Cloud freely. With Nimbits, users can create data points on the Cloud and feed changing numeric, text based, or xml values into them. Data points can be configured to perform calculations, generate alerts, relay data to social networks or can be connected to spreadsheets, websites, and more. Nimbits offers a data compression mechanism, an alert management mechanism, and data calculation on the received sensor data, using simple mathematic formulas.

ThingSpeak [14] is another open source Internet of Things application and API (application programming interface) for storing and retrieving data from Things which uses HTTP over the Internet or via a Local Area Network. With ThingSpeak, users can create sensor-logging applications, location tracking applications, and a social network of Things with status updates. The ThingSpeak API allows for numeric data processing such as time scaling, averaging, median summing, and rounding. The ThingSpeak channel feeds support JSON and XML formats for integration into applications.

[1] http://www.mediateam.oulu.fi/projects/mammoth/?lang=en

Paraimpu [15] aims to allow people to connect, use, compose, and share Things, services, and devices to create personalized applications in the field of the Internet of Things. Users can work with Paraimpu connect sensors, motors, micro-controllers such as Arduino, domestic appliances, lighting and domotics systems, smart-phones, or other systems to talk with the Web. Paraimpu allows users to compose and easily inter-connect and mash-up Things to react with events, environmental sensors, or social activities. Paraimpu is a social tool, and it not only communicates with existing social networks, but also allows users to share their Things with friends. This allows avoidance of waste from buying similar objects for the same purpose.

The iDigi Device Cloud [16] allows users to connect a physical device to the Cloud and use an online Web application for remote access. The iDigi Device Cloud application converts complex device data into simple, useful information concerning anything from refrigerator temperatures falling below a specific threshold, to soil quality. The iDigi Platform is a machine-to-machine (M2M) platform as a service. The iDigi Platform manages the communication between enterprise applications and remote device assets, regardless of location or network. The platform includes the device connector software (called iDigi Dia) that simplifies remote device connectivity and integration. The application messaging engine enables broadcast and receipt notification for application to device interaction and confirmation. The application also has cache and permanent storage options available for generation-based storage and on-demand access to historical device samples.

The SensorCloud™ [17] is a sensor data storage, visualization, and remote management platform that leverages powerful Cloud computing technologies to provide excellent data scalability, rapid visualization, and user programmable analysis. The core features include OpenData API, LiveConnect, FastGraph, and MathEngine. The OpenData API allows users to upload sensor data from any Web-connected source or platform, and download data sets. The FastGraph is a sophisticated, time-series visualization and graphing tool. The LiveConnect feature provides users full access to every function available on their wireless sensor network, from anywhere in the world. The MathEngine allows users to process vast quantities of sensor data in the Cloud, and on the fly.

There are various Things development platforms such as Wiring [18], Sun SPOT [19], mbed [20], or Arduino [21]. Wiring [18] is an open-source programming framework for microcontrollers. Wiring allows writing software to control devices attached to the electronics board, to create all kinds of interactive objects, spaces, or physical experiences of feeling and responding to the physical world [18]. Sun SPOT (Sun Small Programmable Object Technology) [19] is a wireless sensor network (WSN) mote. The device is built upon the IEEE 802.15.4 standard. The mbed microcontroller [20] is a single-board microcontroller with associated tools for programming the device. The current hardware of the mbed microcontroller is based around an NXP microcontroller, which has an ARM Cortex M3 core, running at 96MHz, with 512KB flash, 32KB RAM, as well as several interfaces including Ethernet, USB Device, controller area network, Serial Peripheral Interface BusInter-Integrated Circuit, and other I/O. For example, an mbed application can get an RFID tag to trigger a tweet. Arduino [21] is a popular open-source single-board microcontroller and a descendant of the open-source Wiring platform which is designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open hardware design for the Arduino board with an Atmel AVR processor, and on-board input/output support. The software consists of a standard programming language compiler and the boot loader that runs on the board.

III. THINGS-ENABLED SMART HOME SCENARIO: USE CASE STUDY

The equipment used in our smart home scenario includes appliances, stereos, televisions, toasters, microwave ovens, air conditioners, computing equipment such as PCs, PDAs, mobile phones, small controllers (for lights, curtains, and windows), video intercoms, and sensors (for indoor position, temperature, light, rain, GPS, bodies, smoke, gas, infrared microwaves, etc.). Outdoor sensors are included in the scenario as well. Table 1 summarizes the major use cases depicted in the following scenario.

On an early winter morning at 7 a.m., Dr. Smith wakes up as the background music in the bedroom gently rises, and the curtain slowly opens. As he starts washing his face, the background music in the bedroom automatically stops, and the morning news starts in the bathroom. In the kitchen, bread slices and milk have been heated. When Dr. Smith sits at the kitchen table, the TV in the kitchen automatically comes on and tunes to a previously set program. When he stands up to leave, the TV screen presents his memoranda, schedule, and reminder notes for the day. When he leaves for the office, the TV and air conditioning automatically turn off, and the anti-theft systems such as magnetic door locks and infrared microwave detectors start working.

By 10 a.m., the outdoor temperature sensors, light sensors, and raindrop sensors determine it will be a sunny day. The windows open automatically to bring fresh air into the house. Around 4 p.m., the temperature drops, and the windows automatically close. At 5:30 p.m. the central control system calculates that Dr. Smith will be home in 40 minutes, based on tracking the travel speed of his mobile phone and the traffic status reports. Thus, the air conditioning starts and the temperature is set to his favorite 20 degrees.

Dr. Smith leaves his office at 5:30 p.m. The sensor network monitoring air quality reports lowered air quality along his normal route home, so he selects a different route. As the traffic system reports less traffic on that route, the system estimates the effect on fuel consumption to be minimal.

At 6:30 p.m., Dr. Smith arrives home. The room temperature is very comfortable, the lights come on as he enters the living room, and the curtains close. The computer automatically downloads and opens the Word document which Dr. Smith did not finish at the office, and directs him to the last modified lines. Dr. Smith works on the document for a while, then checks the report generated by the house concerning its energy consumption. He notices that the energy consumption has been normal, as most devices have consumed the expected amounts of energy. The energy share for the fridge has been quite high, so he increases the fridge temperature by 0.5 degrees. He is happy to see that the solar panels and the small wind mill on the roof have actually generated so much energy that his house has sold some energy to the smart grid during the day. However, the report on his daily activity (based on his movements measured by wearable sensors) advises him that he has not performed the amount of physical activity he had planned, so he makes a mental note to walk more.

At 7 p.m., Dr. Smith begins to prepare for dinner. His favorite light music automatically starts in the kitchen. He would like to change the music, so he raises and sways his left hand to a different beat, and cheerful music starts to play. When Dr. Smith sits down for his meal in the dining room, the kitchen lights automatically switch off, while the music continues to play in the dining room. When he sits on the sofa in the living room, the TV presents him with the concluding portion of a program he didn't finish watching yesterday.

At 10 p.m., Dr. Smith steps into the bedroom. The lights there slowly come on, while the lights in the living room, the TV, and the stereo speakers all turn off. As Dr. Smith goes sleep, all his devices work in a sleep state as well. Only the smoke, gas, and door security systems remain alert.

TABLE 1. SUMMARY OF USE CASES.

Time	Use Cases	Description
7 a.m.	Ubiquitous positioning service	Locating people and presenting services
10 a.m. to 5.30 p.m.	Physical-world Web	Ability to monitor and control home objects on the Internet
6.30 p.m.	Synchronizing data and ambient metering	Ability to store and synchronize contents in multiple devices and real-time metering of aggregate power consumption
7 p.m.	Intelligent interaction	Ability to respond to human gestures
10 p.m.	Energy-efficient management	Ability to self-control

IV. THINGS FEATURES AND THINGS APPLICATION CHARACTERISTICS

A. Things Features

The above scenario involves many Things and Things applications. These Things present the following common features:

Sensor Things perceive and transmit data. These Things can collect the data on the environment and information related to it (in this case, Dr. Smith's indoor location, indoor/outdoor temperature, real time traffic situation, air quality, energy consumption, gesture commands, or lighting conditions) and transmit them to a different Things when necessary (such as your mobile phone or your laptop) or to the Internet.

Actuate Things are based on trigger events. Their perceived information automatically triggers corresponding devices. For instance in this case, Dr. Smith's waking up triggers turning on the background music in the bedroom and preparing his breakfast. His leaving home triggers turning off air conditioners and the TV, and turning on anti-theft systems.

Things obtain information from the Internet in a pull/push way, since they are part of the Internet. For instance in this case, the TV automatically retrieves Dr. Smith's calendar, memoranda, and reminder notes for the day. The air conditioner fetches Dr. Smith's arrival time and prepares to start; his car obtains the traffic status and makes a routing decision, etc.

Things interact with each other and assist communication. They interact with each other and exchange information. Things also participate in networking and serve as routing nodes that assist in communication, forwarding data to the endpoint.

B. Things Application Characteristics

In the above scenario, there are many Things applications. First, music plays while the user moves around; the TV automatically tunes to a previously set program and retrieves the user's calendar, memoranda, and reminder notes. Second, outdoor temperature sensors, light sensors, and raindrop sensors produce data frequently, and automatically interact with each other to make windows open or close. Road sensors surrounding the city produce data and collaborate with each other to report real time traffic status. Third, the Things automatically fetch and direct information files to the last modification lines. They meter energy consumption, and present a statistic report while initiating sales of extra energy to the smart grid during the day.

To realize the above scenario involves challenges of integrated computing, big data storage, various development

environments, heterogeneous hardware infrastructure management, security and privacy, as well as the following issues.

- Sensors generate a lot of data that needs to be stored and managed. Usually, embedded memory is quite limited. Utilizing memory cards or computers to store sensor data is an alternative way, but these are still limited in storage capacity and require major efforts to manage.
- Things require Web-based interfaces for data exchange and integration between other applications, so it will be possible that the user can access and control Things anywhere.
- Things require sufficient computational and storage resources to handle large-scale applications on demand.
- Things require sufficient computing resources for real-time processing of heterogeneous data, in order to make critical decisions and provide quick response to the user.
- Things require Web-based platforms for programming, deployment, and for updates without creating downtime.
- Things require automatic formation of workflows, and invocation of services to carry out complex tasks.
- Things require different interaction mechanisms, loosely or tightly coupled, synchronized or asynchronized for complex event processing.
- The system requires interoperability between Things, so that the Things are agnostic from heterogeneous hardware and standards.
- Things require use of IT resources (e. g. computer, storage, and network) on demand in a scalable and cost-efficient way.
- Things need to be built in such a way as to ensure an easy and secure data exchange and users' control, and avoid any risks to their security and privacy.

V. CLOUDTHINGS ARCHITECTURE: CLOUD-BASED INTERNET OF THINGS PLATFORM

What makes the Cloud-based Internet of Things different than conventional Internet of Things is basically the ability to develop, deploy, run, and manage Things applications online via the Cloud. Fig. 1 illustrates the main features of the Cloud-based IoT platform (i.e. CloudThings architecture) and their interaction with the three Cloud computing models of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Figure 1 also specifies our technical solutions to networking Things, interacting Things, and integrating Things with the Cloud.

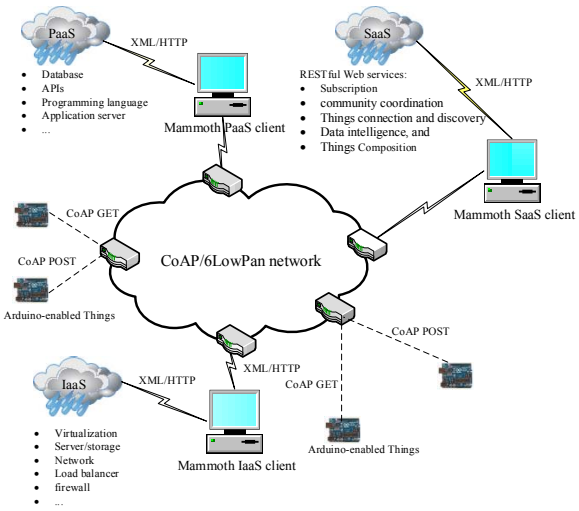

Figure 1. CloudThings architecture: the Cloud-based IoT platform

CloudThings architecture is an online platform that allows system integrators and solution providers to leverage a complete Things application infrastructure for developing, deploying, operating, and composing Things applications and services that consist of three major modules:

- The CloudThings service platform for Things is a set of Cloud services (IaaS), allowing users to run any applications on Cloud hardware. The CloudThings service platform for Things dramatically simplifies the application development, eliminates need for infrastructure development, shortens time to market, and reduces Things management and maintenance costs. The CloudThings service platform offers users unique device management capabilities. It communicates directly with devices and provides storage to collect Things data and transmit Things events. Vast amount of sensor data can be processed, analyzed, and stored using the computational and storage resources of the Cloud. The CloudThings service platform allows sharing of sensor resources by different users and applications under a flexible usage mode.
- The CloudThings Developer Suite for Things is a set of Cloud service tools (PaaS) for Things application development. These tools include open Web service application programming interfaces (APIs), which provide complete development and deployment capabilities to Things developers.
- The CloudThings Operating Portal for Things is a set of Cloud services (SaaS) that support deployment and handle or support specialized processing services including service subscription management, community coordination, Things connection, Things discovery, data intelligence, and Things composition.

A. *Interact with Things Using Constrained Application Protocol (CoAP)*

In the CloudThings architecture, we use CoAP to interact with Things. The CoAP [22] is a specialized Web transfer protocol for use with constrained nodes and constrained (e.g. low-power, or lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as 6LoWPAN often have high packet error rates and a typical throughput of 10s of kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.

CoAP provides a request/response interaction model between application end-points. This supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP easily interfaces with HTTP for integration with the Web, while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments.

CoAP is based on the same client/server model as HTTP, and represents its interaction model in a similar manner. Resources are requested and identified by URIs using the Representational State Transfer (REST) [23] methods of GET, PUT, POST and DELETE. In contrast to HTTP, the CoAP exchanges messages asynchronously over UDP (User Datagram Protocol). The GET method is used to retrieve resources from WSN nodes or telematic devices. The resource is identified by the requested URI. The PUT method is used to modify an existing resource on a sensor node or a telematic device. Both the methods and the requested URI are carried in a confirmable (CON) message which represents the request.

B. *Networking with Things Using 6LoWPAN*

In the CloudThings architecture, we use 6LoWPAN, i.e. IPv6-based Low Power Wireless Area Networks. 6LoWPAN [24] defines message frame formats, fragmentation methods, and header compression techniques required to fit Ipv6/UDP datagrams in the very limited IEEE 802.15.4 frame size. The 6LoWPAN innovations provide IP access to a wide set of networked devices, which, being low-cost, low-power constrained hosts, could not easily benefit from the huge addressing space of IPv6. 6LoWPAN is able to reduce the IPv6/UDP header while maintaining the main functionalities and the size of the addressing space, thanks to a cross-layer optimization approach.

Routing functionalities are provided by the Routing Protocol for Low power and lossy networks (RPL) [25], which are another IETF (Internet Engineering Task Force) solution discussed in the Routing Over Low power and Lossy networks (ROLL) working group. RPL supports different routing path optimizations based on specific objective functions. For instance, high priority packets can be routed to offer low delivery delay, while delay-tolerant traffic can be handled to minimize the energy expenditure or to maximize the network capacity. Another important feature of RPL is its intrinsic scalability with respect to the network density.

C. *Integration with the Cloud Using RESTful Web Services*

There exist two architectural styles for Web services applications: REST [23] and SOA (Service-Oriented Architecture) [26]. Both of these describe the methods for designing and developing interoperable services via Web and design principles. According to [27, 28], SOAs are not well suited for enabling the end-users to create ad-hoc applications; SOAs experience complex functional blocks and service implementations; SOAs are often used to model and realize complex business flows. The RESTful protocol is HTTP, which uses HTTP-similar standardized methods (e.g. GET, PUT, POST, DELETE, etc.) to deal with resources. We adopt REST as architectural style, where Things are modeled as RESTful resources and referred as services

VI. IMPLEMENTATION

This section presents two related prototypes. Fig. 2 shows the smart home application based on a Cloud infrastructure. In the application, the sensors read the home temperature and luminosity from Arduino-enabled IoT things and the Cloud application stores and visualizes them so that the user can view them anywhere, anytime using a Web browser and an Internet connection. Specifically, (1) we use LM35 temperature sensor to sense the home environment temperature; use LDR (light dependent resistor) analog sensor to sense the home light luminosity; (2)we use Ethernet cable to connect Arduino to the Internet; (3)We use HTTP (through a GET and POST request) to send data between Arduino-enabled IoT things and the Cloud application. (4) We use a Cloud service –Google App Engine to host the Cloud application that stores sensor readings and visualizes them. (5) We also use Cloud-based IoT service – Paraimpu to connect Arduino-enabled sensors, and to share the sensor readings with friends.

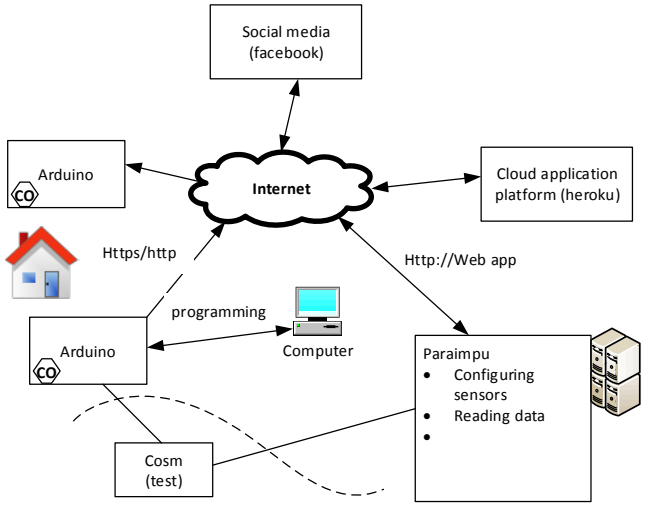

Figure 2. IoT-based smart home scheme

Fig. 3 presents the Cloud architecture to accelerate service composition and rapid application development. We extend the conventional Cloud architecture by inserting a special "Composition as a Service" layer for dynamic service composition. The CM4SC middleware encapsulates sets of fundamental services for executing the users' service requests and performing service composition. These services include process planning, service discovery, process generation, reasoning engine service, process execution, and monitoring, as detailed in [5]. The trial implementation also demonstrates that CM4SC middleware as a service releases the burden of costs and risks for users and providers in using and managing those components.

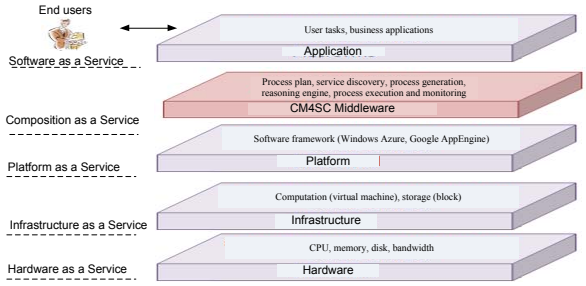

Figure 3. Cloud architecture for dynamic service composition

VII. CONCLUSION

The MAMMOTH project, funded by Tekes (the Finnish Funding Agency for Technology and Innovation), aims to facilitate information exchange and synergic performance between Things and people via global massive-scale M2M (machine-to-machine) networks, and provide M2M automatic metering, embedded Web services, and universal control of electricity or water utilities, etc. The integration of Cloud computing into the Internet of Things presents a viable approach to facilitate Things application development. This paper studies a Things-enabled scenario, and designs a Cloud-based Internet of Things platform – the CloudThings architecture, which accommodates IaaS, PaaS, and SaaS for developing, deploying, running, and composing Things applications. The previous implemented prototypes establish the fundamental developments for approaching CloudThings architecture.

ACKNOWLEDGMENT

This work was carried out through the Mammoth project, which is funded by Tekes, the Finnish Funding Agency for Technology and Innovation.

REFERENCES

[1] Kevin Ashton, "That 'Internet of Things' Thing". In: RFID Journal, 22 July 2009; http://www.rfidjournal.com/article/view/4986/, retrieved by 20 July 2012.

[2] Nicholas Negroponte, "Wireless: Creating Internet of 'Things': A scary, but exciting", http://www.nytimes.com/2005/11/20/technology/20iht-wireless21.html, retrieved by 20 July 2012.

[3] TEKES, "Internet of Things Strategic Research Agenda (IoT--SRA)," 2011, http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2011.pdf, retrieved by 20 July 2012.

[4] F. Kawsar, G. Kortuem and B. Altakrouri, "Supporting interaction with the internet of things across objects, time and space," in Internet of Things (IOT), pp. 1-8, 2010.

[5] Jiehan Zhou, Kumaripaba Athukorala, Ekaterina Gilman,Jukka Riekki & Mika Ylianttila, "Cloud Architecture for Dynamic Service Composition," International Journal of Grid and High Performance Computing, vol. 4(2), pp. 17-31, 2012.

[6] Amazon, "Amazon Web Services," Http://aws.Amazon.Com, retrieved by 20 July 2012.

[7] Google Inc., "Google apps engine," Http://www. Google.com/apps, retrieved by 20 July 2012.

[8] Microsoft, "Windows Azure Platform,"Http://www.Microsoft.com/windowsazure/products/default.Aspx, retrieved by 20 July 2012.

[9] G. Ekaterina, S. Xiang, O. Davidyuk, J. Zhou and J. Riekki, "Perception framework for supporting the development of context-aware Web services," International Journal of Pervasive Computing and Communications, vol. 7, pp. 339-364, 2011.

[10] J. Bungo, "Embedded Systems Programming in the Cloud: A Novel Approach for Academia," Potentials, IEEE, vol. 30, pp. 17-23, 2011.

[11] D. Guinard, C. Floerkemeier and S. Sarma, "Cloud computing, REST and mashups to simplify RFID application development and deployment," in Proceedings of the Second International Workshop on Web of Things, San Francisco, California, pp. 91-96,2011.

[12] Cosm, "https://cosm.com/," retrieved by 20 July 2012.

[13] Nimbits, "http://www.nimbits.com/," retrieved by 20 July 2012.

[14]Thingspeak,"https://www.thingspeak.com/," retrieved by 20 July 2012.

[15] Paraimpu, "http://paraimpu.crs4.it/," retrieved by 20 July 2012.

[16]Device cloud, "http://www.idigi.com/devicecloud," retrieved by 20 July 2012.

[17] sensorcloud, "http://www.sensorcloud.com/," retrieved by 20 July 2012.

[18] Wiring, "http://wiring.org.co/," retrieved by 20 July 2012.

[19] sunspot, "http://www.sunspotworld.com/," retrieved by 20 July 2012.

[20] mbed, "http://mbed.org," retrieved by 20 July 2012.

[21] Arduino, "http://www.arduino.cc/," retrieved by 20 July 2012.

[22] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP) draft-ietf-core-coap-1", http://datatracker.ietf.org/doc/draft-ietf-core-coap/, retrieved by 20 July 2012.

[23] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. dissertation, University of California, Irvine, 2000.

[24] Zach Shelby and Carsten Bormann, 6LoWPAN: The Wireless Embedded Internet. Wiley, 2009.

[25] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", IETF Internet Draft draft-ietf-roll-rpl-19, 2011, http://tools.ietf.org/html/draft-ietf-roll-rpl-19, retrieved by 20 July 2012.

[26] Thomas Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall, 2005.

[27] G. Moritz, F. Golatowski, D. Timmermann and C. Lerche, "Beyond 6LoWPAN: Web Services in Wireless Sensor Networks," Industrial Informatics, IEEE Transactions on, vol. PP, pp. 1-1, 2012.

[28] A. Pintus, D. Carboni and A. Piras, "The anatomy of a large scale social web for internet enabled objects," in Proceedings of the Second International Workshop on Web of Things, San Francisco, California, 2011, pp. 6:1-6:6.