Cloud Space
Web-based Smart Space with Management UI

Anna-Liisa Mattila, Kari Systä, Jari-Pekka Voutilainen and Tommi Mikkonen
Department of Pervasive Computing, Tampere University of Technology, Korkeakoulunkatu 1, Tampere, Finland

Keywords: Internet-of-Things, Mobile Agents, HTML5, Web Applications, Experimentation.

Abstract: The emergence of HTML5 allows more complex applications to be run in browsers. However, these applications need not run inside the browser only. In our previous work we have shown that it is feasible to implement mobile agents with Web technologies, such as HTML5 and JavaScript. These mobile agents can be used to control systems like home automation. In this paper we show how this execution environment can be described as a Cloud Space that provides the users with a new type of multi-device experience to the content and the environment the users need to access and control. Furthermore, we present a new way to control and monitor the Cloud Space through a web application with a 3D UI based on direct manipulation.

1 INTRODUCTION

Modern web applications are systems that use dynamic HTML – HTML, CSS, DOM and JavaScript – for user interfaces and the HTTP protocol as the communication protocol. These systems form the backbone of the current ICT infrastructure. End users can easily access information and services through the Web using the browser without additional installation of specific client software. This commonality and uniformity has simplified end-users' life, and given the browser a central role in all information access and nowadays increasingly often also in entertainment, office applications, and services such as banking.

For users, most of the relevant computing resources and content are located in the Internet. This trend is related to two recent developments: 1) users have multiple devices that they use to access data and services, and 2) users store their content in several cloud-based services. These two aspects are not yet fully supported since information that is typically local – like bookmarks – is often on a wrong device, and data that is stored in cloud-based services cannot be accessed as seamlessly as data stored in user's own devices. Although some services provide a REST API – an architectural style for systems where resources play a major role (Fielding, 2000) – to access the content in a generic fashion, far too often data is only available through the specific service or proprietary application, and it is not possible to create new applications that would benefit from that data.

The increasing amount of computing power in our everyday environment is also an emerging trend. For example, home entertainment systems are increasingly often online, with considerable computing and storage capacity. We are surrounded by several "smart spaces", or systems where our environment is controlled and monitored with computers and software running in them. Cheap computing devices, such as Raspberry Pi (Raspberry PI, 2013), have become widely available and are suitable for research and do-it-yourself (DIY) needs. This enables low-risk experiments and provides evidence that low-cost, Internet connected and powerful computing nodes are coming closer to us. We believe that the most natural way to access these pervasive computing elements are through a browser instead of platform specific installable proprietary solutions which are commonly used in today's smart spaces.

In this paper we describe vision and proof-of-concept implementation of a system called Cloud Space, where all computing resources form interoperable clouds that users can access with any device. Users do not need to care about boundaries between particular services or individual devices available in smart spaces. Still, however, users can maintain their ownership and they are in control of their own content. In other words, data and computing will be completely cloudified for the user without risking users' ownership of the content. In addition, we show how such an environment can be controlled and managed with a virtual 3D environment built using the

Mattila A., Systä K., Voutilainen J. and Mikkonen T..
Cloud Space - Web-based Smart Space with Management UI.
DOI: 10.5220/0004961001370142
In *Proceedings of the 10th International Conference on Web Information Systems and Technologies* (WEBIST-2014), pages 137-142
ISBN: 978-989-758-024-6

same technologies. In fact, the monitoring application could be stored in the same cloud infrastructure as the applications it is presently managing.

The rest of the paper is structured as follows. Section 2 describes the background. Section 3 introduces the concept of Cloud Spaces, and Section 4 describes a manager application for Cloud Spaces. Section 5 addresses related work. Finally Section 6 draws some final conclusions.

2 BACKGROUND

Our work has been motivated by a number of already reported research artifacts. These artifacts will be briefly introduced in the following subsections.

2.1 Interactive Web Applications

Emerging web technologies such as HTML5 (World Wide Web Consortium, 2012) and WebGL (Khronos Group, 2011) have rapidly altered the landscape of web application development. With such technologies, it is feasible to develop interactive applications with web technologies only, with no vendor-specific plug-ins that require separate installation. Even for games, the browser is increasingly often the desired platform due to its convenience – the users never need to install anything except the browser itself.

These interactive applications are changing our perception of browser-based applications. The code that runs in a browser is no longer a simple rendering procedure generated in the server-side. The applications are deployed and updated from the server like a web page, but after that the needs of the applications determine how dependent the execution is of the server. The applications can even be cached in the device so that they can be used in off-line situations. Examples of large web applications include systems like Google's Gmail[1], which consists of a considerable amount of code run inside the browser.

At the same time increasingly powerful libraries have simplified the development of JavaScript applications. These libraries help dealing with browser incompatibilities and JavaScript's not-so-good parts (Crockford, 2008). Furthermore, they support the development of web applications that provide the look and feel that is fundamentally similar to desktop applications. This was shown to be feasible by the Lively Kernel, which provided Smalltalk-like programming environment inside the browser (Ingalls et al., 2008), and Cloudberry (Taivalsaari and Systä, 2012) which showed that all user-visible software of a smart phone can be implemented with dynamically downloaded web-content.

[1]http://gmail.com/

2.2 Evolving Web Architecture

Latest web application trends have brought technologies originating from the browsers to servers, too. For example, Node.js (Node.js, 2013) has lately gained popularity. Node.js allows running JavaScript in the server side, and thus to some extent Node.js allows the execution of the same code in both server and client. This is a core enabler of our HTML5-agents described in the next subsection.

Technologies like Meteor (Meteor, 2013) and backend-as-a-service systems like Firebase (Firebase, 2013) are also changing the web architecture towards cloud-like architectures. In general, these systems provide features that are commonly needed online. Such features include user management, push notifications, and integration with social networking services, all of which are commonly needed in web applications.

2.3 HTML5 Agents

The Web is fundamentally based on mobile code, pages that may contain applications are downloaded from remote locations and evaluated inside the browser. Four paradigms of mobile code have been proposed (Carzaniga et al., 1997):

1. Client-Server where client uses code that is located in another node.
2. Remote Evaluation where client sends execution instructions, e.g. SQL queries, to another node.
3. Code on Demand where code is downloaded to the client for execution. HTML5 applications are widely used examples of code-on-demand.
4. Mobile Agent where code together with internal state of the application is moved to other node for execution.

The first three paradigms are regularly used in Web applications; the fourth paradigm is connected to cloud browsing paradigm (Taivalsaari et al., 2013) at least indirectly. In the following, we focus on the fourth paradigm, i.e. mobile agents.

In our previous work, we have designed an agent architecture (Systä et al., 2013), where agents can travel between hosts which can be either browsers or web servers. When running inside browsers these agents act like any Web application, but when located in a server, they are run in headless mode without a

user interface. The agent can move between servers and browsers and thus toggle the mode between headless mode and UI mode. Furthermore, we support multi-device usage – the browser instance that pulls an executing agent can be different from the browser instance that had originally pushed the agent to the server. The internal state of the application is an important part of a mobile agent. HTML5 agent's state needs to be serialized when the agent is moved from a location to another. Thus, the agent needs to be written so that serialization of the relevant parts of the state is possible. Our design provides support for such serialization. During agent's life-cycle the agent may visit several browsers and several agent servers. A sample life cycle is presented in Figure 1.

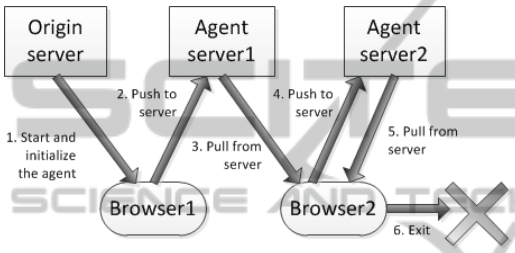

Figure 1: Life-cycle of a HTML5 agent (Systä et al., 2013).

In Figure 1, an agent is started by Browser1, when the agent is downloaded from its origin server (Step 1). In this phase the agent is initialized and the execution begins. Since the agent executes in a browser it has a user interface. In Step 2, the agent is pushed to an agent server. This means that the agent server gets the internal execution state of the agent and the application code (actually a URL to the code). The agent can continue the execution in the server. In Steps 3-5, the agent moves from one environment to another, but preserves its internal state and continues execution from where it left off. Finally, the execution is terminated in Step 6.

In our present implementation, the life cycle shown in Figure 1 has been modified so that agents can move between server and client but also between two servers. This liberates applications to travel freely between different computing nodes included in the system.

2.4 Agents for Web-of-Things

In (Järvenpää et al., 2013) we proposed using mobile agents in the context of Internet-of-Things (IoT). The approach was based on the fact that agent servers can be instantiated in small embedded computers and the mobile agents can move between different devices, and if necessary it is also possible to clone agents to create more instances. One of the possible application areas we discussed in (Järvenpää et al., 2013) was home automation that goes beyond remote control when the agents can be run in embedded devices installed in the physical environment. In that case an intelligent agent can work on behalf of the user and implement even complex strategies to optimize energy consumption and user comfort.

There are approaches that are based on uploading and remote evaluation of code in a "thing". For example, MoteLab (Werner-Allen et al., 2005) is a test bed for sensor networks, where developers can upload executable Java to a device. Somewhat similar system is Kansei (Ertin et al., 2006) (later refactored to KanseiGenie) where developers can also create jobs to execute sensor applications. Our system can also be used in a similar way and from similar motivations. However, in our system the uploaded code is Web content and we can upload an executing agent with its internal state – i.e. our system is the paradigm of mobile agents.

Maybe the most similar approach to us is the mobile agent framework proposed in (Godfrey et al., 2013). It provides nodes in heterogeneous device networks with a way to communicate and co-operate. The system is based on Java-based AgentSpace (Silva et al., 1999) mobile agent platform. For us the use of Web technologies is essential since it enables leveraging the power of the web development ecosystem in application development (Systä et al., 2013). Furthermore, our agents have been designed to work a part of the Cloud Space explained in Section 3.

Since our mobile agents are based on Web technologies – HTML and JavaScript – they integrate well in Internet-based infrastructures. Moreover, normal browsers can be used to access and control the agents.

3 CLOUD SPACE

A key concept of our work is computing and content service called *Cloud Space*. Users have their own Cloud Spaces – in essence private clouds – in which all content is stored. Cloud Space includes the following three parts that relate and depend on each other:

1. A data solution that provides a uniform access to all content. Unlike typical cloud service, Cloud Space does not enforce service-specific silos or limit what kind of content is uploaded to the service. All the content is synchronized automatically between Cloud Space and devices and applications using the content.
2. A system that stores interaction (browsing) sessions so that they can be later continued in another

device.

3. The software infrastructure for mobile agents.

3.1 Concept

Cloud Spaces can be virtualized. For instance, a single computing resource can host several Cloud Spaces. However, like in cloud computing, the Cloud Spaces appear to both users and applications as a single service. The Cloud Space hosts arbitary amount of *contexts*, which represent physical locations with devices like a kitchen, living room and mobile phone. Each context is a view to one or more Cloud Spaces with a user interface optimized for different screen sizes and for the needs of the context. Mobile phone context would need an easy access to lot of content in user's personal Cloud Space and Cloud Space of the current location but with simplistic UI since most likely, the screen size is small. Additionally to the context user interfaces, the admin interface needs its own UI. The admin UI handles creation of new users, new contexts and so on.

An example configuration of Cloud Space has been presented in Figure 2.

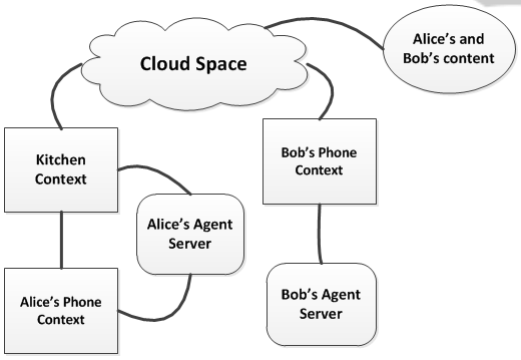

Figure 2: Concept of Cloud Space.

It should be noted that Cloud Space contexts can represent both physical and virtual entities. Furthermore, when user is in some physical space, for example in a room, she can access both her own personal Cloud Space and the Cloud Space of the room.

3.2 Implementation

Cloud Space is partly implemented as a proof-of-concept system. User management and context creation are implemented and they are available in the admin user interface. Currently contexts are bound to physical locations and agent servers can be assigned to a context. The design is implemented in Node.js, mainly to leverage the existing agent server implementation. Hosting the user content will be exported to backend-as-a-service platform because synchronizing data across multiple databases is out of scope of the research.

4 MANAGEMENT UI IN 3D

As already pointed out, mobile agents can be used for managing and monitoring devices in the spirit of Internet-of-Things (Järvenpää et al., 2013). However, in our previous publications, (Systä et al., 2013), (Järvenpää et al., 2013) the management of the agents has been left as future work. In the following, we address the management of IoT environment where Cloud Space contexts and agents are located in the physical space.

4.1 Concept

The concept of the manager application is to provide a tool for a user to manage her Cloud Space contexts related to physical spaces, e.g. to a living room. Using the manager application the user can connect to the Cloud Space, select context and toggle between contexts. The user can monitor agent servers running in a context and move agents from a server to another server and to another context.

Because Cloud Space contexts are bound to physical spaces 3D user interface for the manager is a natural choice. A context can be visualized as a 3D model of the physical space to which the context is related. Resources related to the context and navigation widgets can be visualized as 3D objects.

Cloud Space and mobile agents use browsers as the front end. Similarly the monitor application is based on web technologies and runs on the browser.

4.2 Implementation

We are currently working on a proof-of-concept implementation of a Cloud Space context manager using WebGL and WebWidget3D (Mattila and Mikkonen, 2013) system. WebWidget3D is a 3D widget library that provides tools for creating 3D widgets and designing associated interaction. For actual rendering, WebWidget3D uses Three.js (Three.js, 2014) 3D engine for WebGL.

While the use of WebGL enables the creation of seemingly arbitrarily complex graphics and interactions, we simply aim at demonstrating the feasibility of our approach to the control and management tasks.

Consequently, the implementation effort has been invested in the creation of a simple model that can be easily understood and interacted with.

In the present implementation the user can manage her Cloud Space contexts using a direct manipulation 3D UI. Instead of using 3D models to visualize Cloud Space contexts, panoramic image spheres inside a 3D world are used. The agent servers and agents as well as navigation widgets are visualized using 3D objects. All user interaction is done using direct manipulation of 3D content. The user can move in the 3D world according to simple fly paradigm.

The manager consists of three views:

1. Login View where user logs in to her Cloud Space. This view is simple HTML form in 2D.
2. Context Selection View shown in Figure 3.
3. Context Management View shown in Figure 4.

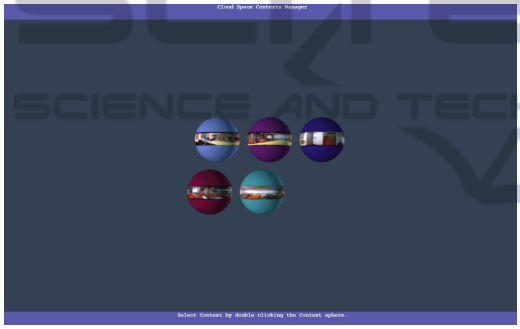

Figure 3: Context selection view.

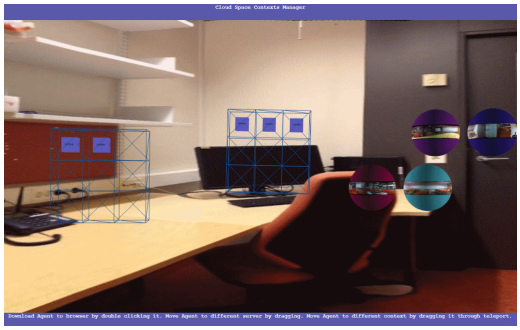

Figure 4: Context Management View.

When the user connects and logs in to her Cloud Space the manager fetches her contexts and moves to context selection view. The user can select the context to manage by double clicking a context sphere.

When the user has selected the context to monitor the Context Management View of the selected context is shown. The view consists of the servers related to the context visualized as grids inside the context sphere. Boxes inside grids are representations of agents. The user can load an agent to her browser by double clicking an agent box. Moving an agent to another server can be done by dragging the agent to another grid and dropping it there.

Inside the Context Management View the Selection View is also visualized. The user can toggle between Context Management Views using the Selection View. The Selection View inside a Context Management View is shown in right side of Figure 4. The user can also drag agents from a context to another. Moving agents are visualized in Figure 5.

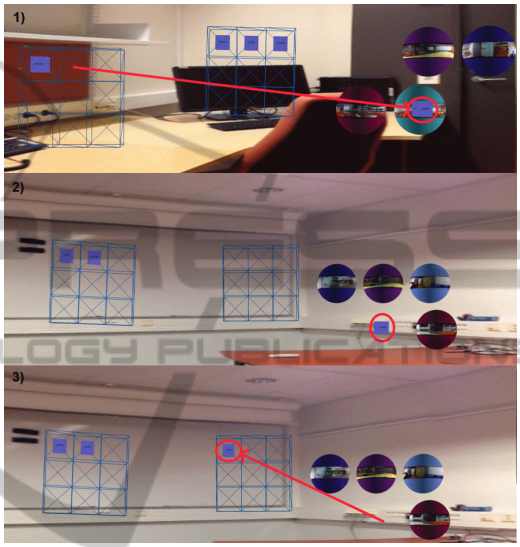

Figure 5: Moving agent from a Context to another Context.

In the first fragment of Figure 5 (marked with number 1) the user has dragged the agent on top of the context which she wants to move the agent. When she releases the agent the Management View changes to the context she chose. This is visualized in the fragment 2 of Figure 5. Finally the user can drag the agent to the server in the context and the agent is moved there (fragement 3 in Figure 5).

5 RELATED WORK

As far as we know our HTML5 mobile agents is a unique application of HTML technology. However, mobile agents as such are an old invention of technologies and concepts (Carzaniga et al., 1997). Similarly, Cloud Space is a unique combination, although parts of it have been realized by commercial systems like iCloud by Apple, which implements some parts of the Cloud Space's idea but is limited to a vendor-specific "silo".

Visualization of the agent configuration is implemented, for example, in MoteLab (Werner-Allen

et al., 2005) where nodes in a sensor network can host Java-based agents. In MoteLab a Maps Page shows the sensor network with connections, but the visualization is two-dimensional only and does not include management operations. Some management aspects have been included in the WebIoT architecture (Castellani et al., 2012), where a heterogeneous device set can be visualized and controlled through an extensible user interface. Similarly to our approach the UI of WebIoT is based on the Web and runs in a browser. However, WebIoT does not use 3D to show the UI and does not introduce direct manipulation.

6 CONCLUSIONS

In this paper we have introduced the concept of Cloud Spaces and a proof-of-concept 3D interface for managing Cloud Spaces in the IoT context.

The concept of Cloud Spaces provides personal clouds for user, but it is more than data storage. A Cloud Space can also host agent servers and a Cloud Space context can be connected to a physical space. This makes it possible to use Cloud Spaces together with HTML5 agents to build smart spaces for e.g. home automation.

The 3D UI for managing contexts inside a Cloud Space shows how a browser can host interactive 3D user interface for monitoring and managing smart spaces. The UI provides functionality, e.g. for moving agent from server to another server, with direct manipulation in the 3D context. Exploring the most feasible 3D visualizations for contexts, agent servers and agents in a smart space are at this point left as future work.

REFERENCES

Carzaniga, A., Picco, G. P., and Vigna, G. (1997). Designing Distributed Applications With Mobile Code Paradigms. In *Proceedings of the 19th international conference on Software engineering*, pages 22–32. ACM.

Castellani, A. P., Dissegna, M., Bui, N., and Zorzi, M. (2012). WebIoT: A Web Application Framework for the Internet of Things. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 202–207. IEEE.

Crockford, D. (2008). *JavaScript: The Good Parts*. O'Reilly.

Ertin, E., Arora, A., Ramnath, R., Nesterenko, M., Naik, V., Bapat, S., Kulathumani, V., Sridharan, M., Zhang, H., and Cao, H. (2006). Kansei: A Testbed for Sensing at Scale. In *Proceedings of the 4th Symposium on Information Processing in Sensor Networks (IPSN/SPOTS TRACK*, pages 399–406. ACM Press.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-Based Software Architectures*. PhD thesis, University of California.

Firebase (2013). Web page of firebase technology. Technical report. https://www.firebase.com/. Last viewed 31.12.2013.

Godfrey, W. W., Jha, S. S., and Nair, S. B. (2013). On a Mobile Agent Framework for an Internet of Things. In *Proceedings of the 2013 International Conference on Communication Systems and Network Technologies*, CSNT '13, pages 345–350, Washington, DC, USA. IEEE Computer Society.

Ingalls, D., Palacz, K., Uhler, S., Taivalsaari, A., and Mikkonen, T. (2008). The Lively Kernel a Self-Supporting System on a Web Page. In *Self-Sustaining Systems*, pages 31–50. Springer.

Järvenpää, L., Lintinen, M., Mattila, A.-L., Mikkonen, T., Systä, K., and Voutilainen, J.-P. (2013). Mobile Agents for the Internet of Things. In *System Theory, Control and Computing (ICSTCC), 2013 17th International Conference*, pages 763–767. IEEE.

Khronos Group (2011). WebGL Specification. Technical report. http://www.khronos.org/registry/webgl/specs/1.0/.

Mattila, A.-L. and Mikkonen, T. (2013). Designing a 3D Widget Library for WebGL Enabled Browsers. In *proceedings of the 28th Symposium On Applied Computing*, volume 1, pages 757–760. ACM.

Meteor (2013). Web page for meteor technology. Technical report. https://www.meteor.com/, Last viewed 31,12,2013.

Node.js (2013). Web page for document and download of node.js technology. Technical report. http://nodejs.org/. Last viewed 31.12.2013.

Raspberry PI (2013). Web page of raspberry pi. Technical report. http://www.raspberrypi.org/, last viewed 31.12.2013.

Silva, A., Silva, M. M. d., and Delgado, J. (1999). An Overview of AgentSpace: A Next-Generation Mobile Agent System. In *Proceedings of the Second International Workshop on Mobile Agents*, MA '98, pages 148–159, London, UK, UK. Springer-Verlag.

Systä, K., Mikkonen, T., and Järvenpää, L. (2013). HTML5 Agents - Mobile Agents for the Web. In *WEBIST*, pages 37–44.

Taivalsaari, A., Mikkonen, T., and Systä, K. (2013). Cloud Browser: Enhancing the Web Browser With Cloud Sessions and Downloadable User Interface. In *Grid and Pervasive Computing*, pages 224–233. Springer.

Taivalsaari, A. and Systä, K. (2012). Cloudberry: An HTML5 Cloud Phone Platform for Mobile Devices. *Software, IEEE*, 29(4):40–45.

Three.js (2014). Web page of three.js 3d engine. Technical report. http://threejs.org/, last viewed 7.1.2014.

Werner-Allen, G., Swieskowski, P., and Welsh, M. (2005). Motelab: A Wireless Sensor Network Testbed. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 68. IEEE Press.

World Wide Web Consortium (2012). HTML5 Specification, candidate recommendation. Technical report. http://www.w3.org/TR/html5/.