Author(s): Mohit Sethi, Maria Lijding, Mario Di Francesco and Tuomas Aura

Title: Flexible Management of Cloud-Connected Digital Signage

Year: 2015

Version: Post print

Please cite the original version:

Mohit Sethi, Maria Lijding, Mario Di Francesco and Tuomas Aura. Flexible Management of Cloud-Connected Digital Signage. In Proceedings of the 12th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC), Beijing, pp. 205-212, ISBN 978- 1-4673-7211-4, August 2015. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.52

Flexible Management of Cloud-Connected Digital Signage

Mohit Sethi*†, Maria Lijding‡, Mario Di Francesco†, Tuomas Aura†

*NomadicLab, Ericsson Research, Finland
†Aalto University, Finland
‡Smart Signs Solutions, The Netherlands
mohit.sethi@aalto.fi, lijding@smartsigns.nl, mario.di.francesco@aalto.fi, tuomas.aura@aalto.fi

***Abstract*—Electronic displays and digital signage have become ubiquitous over the years, and we view or interact with them on a daily basis. With the emergence of cloud computing and web technologies, electronic displays can be connected directly to cloud services from which they retrieve both configuration commands and HTML5 content. Cloud-based management makes it easy to update the displays, and the use of standard web languages enables rapid development of new dynamic applications. However, the remote management aspects and scalability from small personal deployments upwards have received limited attention in the literature. To this regard, our major contributions are the collection of requirements from actual digital signage deployments and the design of a system that encompasses all the different phases involved in the related lifecycle. We specifically design, implement and evaluate a flexible and user-friendly digital signage management system.**

***Keywords*-Digital signage; electronic displays; requirements; management; lifecycle; web; HTML5; cloud.**

I. Introduction

Electronic displays and digital signage are a perfect example of technologies that have disappeared into our daily lives, following Mark Weiser's vision of ubiquitous computing [1]. In fact, they have become such a pervasive part of our environment and surroundings that we are no longer conscious when we use them. Indeed, public, semi-public and private spaces – such as shops, airports, streets, offices, classrooms and homes – are increasingly relying on digital signs for disseminating information. Depending on the context of their deployment, the content may vary from official information to advertising and from entertainment and art to personal communication. State-of-the-art technologies such as digital paper [2] and wireless charging [3] are seeing rapid commercialization that is not only helping to push down the cost, but also broaden the range of display sizes and improve their power efficiency. One typical application is information displays at airports and other transport stations where central management of the display content ensures that it is always up to date. Yet the same locations are still full of printed signs and advertisements, many of which would benefit from being digitalized, dynamically updated and re-purposed, and managed over wireless networks. It can be expected that many of them will soon be converted to an electronic format.

The methods used for fetching and showing content on these ubiquitous electronic displays have evolved over the years. Large screens wired to a general-purpose microcomputer have been superseded by network-connected media players attached to displays. The current and the future generation of ubiquitous smart displays have direct connectivity to the network and to the cloud from where they fetch the content. The related architecture is illustrated in Figure 1. These displays are uniformly IP-based and mostly use WiFi or similar wireless standards for network access. Using existing or shared multi-purpose networks reduces the deployment and maintenance costs. Smart displays do not need an external media player or computer, which further lowers hardware costs and the related energy consumption. The displays are moving from proprietary player technologies such as the Adobe Flash platform to standards-compliant web browsers that show rich HTML5 content. While connecting displays directly to the cloud has been investigated by the research community in the past [4, 5], only recently have cloud-connected displays started to see commercial deployment [6].

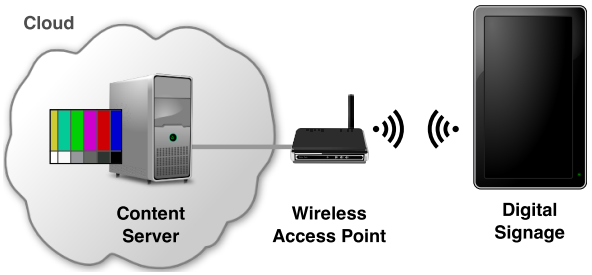


Figure 1: Cloud-based wireless displays

In this article, we build a remote management system for such cloud-connected displays. Our goal is to design a simple and easy-to-use management system with focus on users and staff that have limited knowledge of technology and on deployments that scale from a single display to thousands. The management system is implemented as a cloud application that can be made available as a service. We believe that, as these new electronic displays replace functions previously served by paper and ink, even bulletins, posters and sticky notes will move to a digital format. Therefore, our goal is to allow individuals to install and manage any number of displays with minimal administrative

overhead and no special training. Even though there have been some commercial deployments of management systems for digital signage, there has been limited research into the actual requirements and challenges faced by an average user of these management systems. To this regard, our major contributions are the collection of requirements from actual digital signage deployments and the design of a system that encompasses all the different phases involved in the related lifecycle, with focus on monitoring and management.

The rest of the article is organized as follows. Section II overviews the relevant related work. Section III presents our requirement analysis and the main design principles behind our solution. Section IV details our implementation and Section V discusses some other aspects worth mentioning. Lastly, Section VI draws some concluding remarks.

II. Related Work

Several network management protocols, such as the Simple Network Management Protocol (SNMP) [7], exist and could be employed for managing displays. However, these protocols were designed for network administrators to monitor, track and control network nodes. For instance, SNMP provides read and write functions to re-configure IP addresses, collect information on how much bandwidth is being used, store error reports into a log, and many other configurable options. In order to perform these tasks, SNMP relies on a Management Information Base (MIB) containing structured objects with unique identifiers for every network node that is managed. Needless to say, these parameters are not something that an everyday owner of ubiquitous displays would be interested in tweaking. An average user does not manage personal devices such laptops, tablets, smart watches for network parameters and therefore it is natural to assume the same for display devices.

One of the most important aspects of the display that users want to control is the content, i.e. what is shown on the display at any given time. Clearly, existing Content Management System (CMS) tools for the web are not appropriate. Many of them are too complex for the typical user audience. Back in 2006, Storz et al. [8] had observed that managing content for ubiquitous displays is a considerable task for which tools available to them were poorly suited. They did not find a CMS suitable for ubicomp deployments, even after examining systems from both the broadcast and web communities.

Sugiura et al. [9] designed a distributed digital signage system for multimedia content called InfoShare. Specifically, they provided a relatively simple management interface allowing users to select the location and content that needs to be shown on different displays. The user can choose from plain text, XAML, image or video content and can also specify the time duration for which the content will be displayed. Once any content gets published, the users can remove or edit it at any time through the management interface. While we were indeed inspired by their solution, we found that it was lacking options for remote access and identified all the displays in the system with integer numbers. Such an approach is not ideal as the users would need to remember the unique identifier for every display that they own to perform any management tasks. Clinch et al. [10] discussed the requirements, the design principles and the implementation of an application store for open display networks. Even though their solution is web-based and capable of supporting several applications, the focus of the work is on scheduling the content shown by the displays as well on developer support, including analytics and billing. In contrast, we specifically targeted the configuration and management of digital signage during their lifecycle.

Lindén et al. [11] developed a web system for managing the screen real-estate. They implemented a framework that facilitates dynamic partitioning of the area on a display into several virtual screens assigned to multiple concurrent web applications. The authors use a resource manager to feed in the content and a layout manager that decides how the multiple content applications should be displayed on the screen. However, the authors rely on an external digital signage system for creating and uploading the content playlist to the resource manager. Similarly, Kuikkaniemi et al. [12] pre-loaded content to be shown on walk-up-and-use displays through a special component. While the details are not elaborated, it appears that these systems are designed for infrequent re-configuration and content updates that are handled by professionals skilled in performing these tasks.

Heikkinen et al. [13] have discussed aspects related to remote monitoring and management in the context of a long-term deployment of pervasive displays. In particular, their system employed the Nagios software to collect the state of different resources at the displays, including memory and CPU usage as well as network connectivity. However, such a solution was found more useful for profiling and debugging purposes rather than for detecting actual failures. Similarly, Friday et al. [14] have presented the lessons learned from a long-term experimental deployment of public displays, including maintenance. However, their focus was more on interactive applications rather than on the digital signage scenario we target in this paper.

There are also several commercial digital signage management systems that are currently available on the market. For instance, the BroadSign digital signage management system [15] provides templates for creating content and automatic scheduling of content playlists. They also provide additional functionality of adding or removing content based on external conditions such as weather, traffic status and GPS location. A wide range of formats for audio, video and static content is supported. Finally, their system can also split the display area into many zones similar to the Lindén et al. screen real-estate management framework. The BroadSign management system represents a typical

commercial offering with extensive features, most of which can be troublesome for an average user.

III. System Design

In the following, we present the motivation for and the design principles of our solution. We start by analyzing the requirements we have collected from actual deployments of digital signage in different scenarios. We then illustrate the different phases of a digital signage lifecycle, with specific reference to aspects of administration and management. We finally detail our design choices for the different phases of the deployment.

Before proceeding further, we would like to recall that our management system is tailored to displays that are small, inexpensive and directly connected to the wireless network and to the cloud through a wireless communication technology. These displays use a standard web browser engine for rendering HTML5 content. Even though electronic displays may be typically owned and managed by a single user, in the rest of this article we will rather focus on enterprise settings as they are the most demanding in terms of their required functionalities.

A. Analysis of Requirements

We have collected requirements about digital signage systems from different deployments. They included not only experimental system used at universities and research centers, but also solutions deployed in enterprises including hospitals, business parks, facility companies, and government agencies. Some of the requirements were explicitly pointed out by the stakeholders, others emerged from the product support of long-term digital sign deployments. We summarize the most important ones for our study below.

Flexibility. Digital signage deployments are long-lasting, with a target lifetime of several years. During this timeframe, there may be different changes to the organization adopting the system, ranging from a change of brand, to renovations of the facilities, or even relocation to a different area. The digital signage system should be flexible in handling such organizational and physical changes, in the sense that it has to be easily reconfigured. Here, reconfiguration does not involve only content or software updates, but rather encompasses the entire lifecycle of a digital signage. For instance, a wireless display deployed in one building may need to be moved to another one, possibly provided with a different network configuration, following a company relocation. Thus, the display needs to be removed from the first physical location and network, moved to the target ones and re-configured with the new operating parameters.

Customization. Organizations have their own peculiarities and features that need to be reflected in the digital signage system. Among them, each device can identify spaces with different levels of granularity. For instance, digital signs can be associated with desks in offices adopting shared spaces; they can be linked to a bed in a hospital. The information system of such an environment may have a peculiar identification system that has to be followed by the digital signage system.

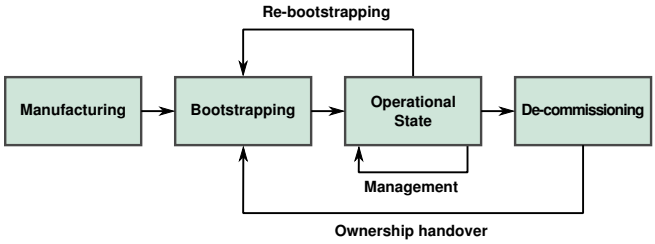


Figure 2: Lifecycle of a digital sign

Direct control. After introducing a digital signage system, many organizations have realized that it has become an asset for their services and workflow. Consequently, they are becoming more and more reluctant to outsource the management of the displays to external companies. They are willing, instead, to have their own employees[1] to administer their system. In order to do so, however, the management and monitoring functions have to be user-friendly, thus easy to use for untrained and non-technical personnel.

B. Lifecycle of a Digital Signage

Every display in a digital signage system has an associated lifecycle, as illustrated by Figure 2. Such a lifecycle begins when the display is *manufactured*. Thereafter, the display is installed and commissioned for network access by an installer. Depending on the deployment scenario, the installer may be the manufacturer, the end user, or some third party (e.g., a facility manager or janitor). In particular, our system design and the underlying protocols allow users to install their displays themselves.

After the display has been powered on, it enters the *bootstrapping* phase during which it is initially configured and associated (i.e., registered) to a certain cloud service, namely, the content provider. Once the display has been bootstrapped with a network and a cloud connection, it enters the *operational* state and is under the control of the owner. During this phase, the owner performs various management tasks, such as updating what is shown on the display. The display remains in the operational phase until it is finally *de-commissioned* and removed from the network. However, it is possible that the removed device is re-commissioned in a new network under a different owner, thus, starting the lifecycle back from the bootstrapping phase. Since the bootstrapping and operational phases are inter-related and are performed by the users themselves, we design the management system by taking into account both these phases.

[1]For instance, one company – after using our digital signage solution for two years – has explicitly asked us to provide them with additional modules, so that its employees could configure and manage the system themselves.

Display Name & Info	Image	Display ID	Time of Conifguration	Location	Current Content	Remote Monitor	Modify data
Ben display corridor		5c:26:0a:1f:ae:d0	2014-08-20 10:30:13	Espoo, Finland	Link/URL	https://dms.ericsson.com /vnc_auto.html?dmacd=5c:26:0a:1f:ae:d0	Modify data
James corridor HD display		1c:65:9d:c6:1c:77	2014-08-20 16:05:14	Espoo, Finland	Link/URL	https://dms.ericsson.com /vnc_auto.html?dmacd=1c:65:9d:c6:1c:77	Modify data
Alex corridor LED display		00:1b:77:20:af:b0	2014-09-07 15:34:48	Espoo, Finland	File	https://dms.ericsson.com /vnc_auto.html?dmacd=00:1b:77:20:af:b0	Modify data

Update

Figure 3: Management interface

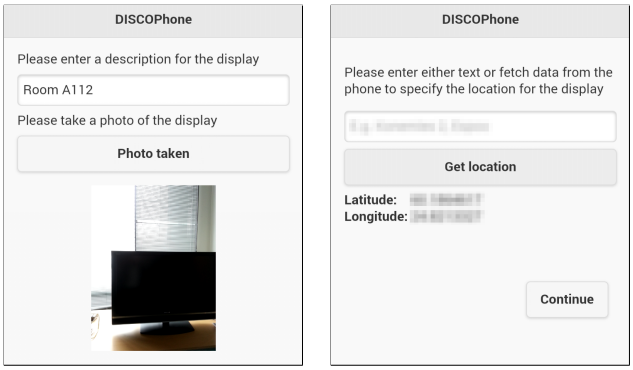
C. Design Considerations

To explain our system design and the rationale behind it, we pick a concrete use-case scenario: Soili is a university secretary and she is taking care of a department corridor containing electronic displays as door signs for employees. Soili is responsible for managing these displays and for updating their content according to the requests from the employees. For instance, the displays may be used to show the calendar of employees, their current status (e.g., busy or away, and eventually their location), or an advertisement for an open research position. We next detail the relevant phases of the digital signage lifecycle.

1) Display Registration: Going back to our use case, let us assume that a new employee has joined the department and that Soili has ordered a new display accordingly. After the display is delivered to the department, the janitor hangs it on the wall next to the office door and powers it up, eventually by connecting the device to a power supply if necessary. Once the display is on, Soili provides it with the necessary configuration parameters for connecting to the Internet and to the intended cloud service provider. Such a cloud service will provide the management interface for the display. Then, Soili may also select the initial content to be shown by the display.

While a wired display could automatically connect to the Internet, wireless device requires some preliminary configuration to obtain connectivity. This configuration may only need a 2D tag such as a Quick Response (QR) code [16]. Alternatively, the SSID and the password of the appropriate wireless access point may be entered manually.

Once the display has Internet access, it will still need to authenticate and register itself with the cloud service. This again may be done manually by pointing the web browser to the correct Uniform Resource Locator (URL) and entering the username and password or automatically by scanning a QR code [17]. Our management system is agnostic to the bootstrapping solution that is used for providing the displays with all necessary security and network credentials. However, we expect that, during this registration process, the unique identity (i.e., the MAC address or a cryptographic public-key identity) of the display is communicated to the management service running in the cloud. As a part of this bootstrapping process, the user is also encouraged to enter some contextual information about the display that is useful for management purposes. The relevant contextual metadata include a textual description of the display, its purpose and location, and an illustrative photo. As an example, Figure 4 shows the interface presented to the user by the bootstrapping solution realized in [17]. Finally, when this bootstrapping process is over, the display shows the default page received from the cloud service and the user is able to redirect it to the actual content.



(a) (b)

Figure 4: The display configuration wizard in [17]. The user can enter (a) a textual description and a picture of the display as well as (b) its location as part of the bootstrapping process.

2) Management Interface: The management interface is web-based and is realized as a scalable cloud service where users and organizations can sign up for accounts. The number of displays belonging to each user account is currently small but we expect it to grow to thousands and beyond for some users. The same service should eventually be able to scale from small experimental and home deployments to organizations with very large numbers of displays. The focus of our current implementation has been on finding the minimal set of universally required management features.

Continuing on the reference use case, Soili returns to

her office and logs into the online management dashboard shown in Figure 3. Every display is uniquely identified by its MAC address or a cryptographic key fingerprint. As it is difficult or even impossible for a user to remember such information, we exploit various of metadata to help users to recall and identify the display. As mentioned earlier, this metadata includes a textual description and a picture of the display. In our example, Soili has already entered a descriptive text for three displays during the bootstrapping phase by adding the employees' names to the description of the corresponding displays. She has also taken a picture of the devices, which makes it easier for her to identify and distinguish the displays based on visual memory. This information is readily shown in the dashboard.

As for location, the user can geotag the display while bootstrapping it. Even though this may not help in locating the displays deployed indoors, geographic coordinates may be useful in distinguishing between different buildings, e.g., on a university campus. Our cloud backend can also guess the location of a display purely based on its IP address, thereby not requiring any explicit action from the user. The location information is provided in the management interface as a link to a location service such as Google Maps.

The user can modify the metadata associated to a certain display at any time from within the management interface. For instance, Soili can update the textual description after the display has already been bootstrapped and a typing error is discovered. This feature is useful since users can make mistakes during the bootstrapping process and may need to update the information later.

3) Content Selection: Directing displays to the desired content is perhaps the most important feature for any display management system. As apparent from our management interface, the user is provided with a simple dropdown menu from which pre-configured content can be selected. The default choices include weather, news, as well as other URLs and files. The user can choose an arbitrary URL or a file to be shown through the advanced configuration button next to the dropdown list in Figure 3. When a user uploads a new file that needs to be shown on a particular display, our cloud backend saves this file as a new resource and assigns it an addressable URL in the REST API. The display always points its browser to the most recent URL received from the cloud backend.

Consequently, a display can be instructed to fetch its content from any web page or web application. The content is typically hosted on a cloud server other than the management one. We use standard HTML5 for rendering content to the web browser engine on the display, and most of the well-known video, audio and textual formats are already supported by modern browsers. Moreover, while our current system is aimed mainly at showing static or dynamic content, it is technically possible for a web application to exploit interactive features such as a touch screen or a camera built into the display. In our use case, Soili configures Ben's corridor display to show his public calendar; she selects an out-of-office notice for Alex's display, instead.

4) Remote Monitoring: Since we rely on HTML5 content that is directly rendered by the smart display, the related software (i.e., a modern web browser engine) may need to be updated or may crash, thus requiring a manual restart. Therefore, it is necessary to give users a remote monitoring option to perform such tasks. All the related functions should be web-based and should work without the need for installing browser add-ons. Our management interface provides low-level access to the display operating system through a URL unique to each registered display. The user can simply click on it and access the remote operating system to perform simple management operations such as restarts and software updates.

It is worth emphasizing that it is of paramount importance to perform monitoring at different levels. Local network connectivity, including correct initialization of IP addresses and the reachability of the default gateway, is the first step to ensure proper functioning. Once local networking is successfully tested, remote connections such as Secure Shell and Virtual Private Network need to be verified. Besides, the correct functioning of the application running at the digital sign has to be ensured at all times. This can be accomplished by periodically checking that the Internet browser is active in full-screen mode and is actually running in the foreground. Detected issues should be presented to the user in an easy-to-understand language and well-defined procedures should be provided to identify the actual problems. The availability of remote connection also allows external actors (such as a helpdesk) to diagnose and solve the problems.

In the less common scenario in which the display has completely lost connectivity and cannot be remotely accessed any more, the user will have to reset the display by pressing a physical button (e.g., the power button). In extreme cases, the display may have to be registered afresh. Nevertheless, it is important to minimize the need for on-site maintenance, and we therefore implement remote control at different levels, from URL redirection to remote desktop and secure shell access. In closed networks, remote hard resets can also be accomplished, especially if the display is deployed with Power over Ethernet.

IV. Implementation

Our implementation involves software running on the display and in the cloud. We utilize several different tools and programming languages for implementing the different aspects of our management system. The related details are explained next.

A. Bootstrapping

For bootstrapping the displays and initiating a connection with both the WiFi network and the cloud service, we used

the solution previously developed by us [17]. Accordingly, the display shows a dynamic QR code which, when scanned by the user with a camera phone, allows automatic configuration of the wireless network and establishes a secure connection with the cloud service. This is accomplished by a long-term trust relation that is configured between the cloud service and the wireless access network.

We employed Linux-based devices as the smart displays, and we have extended the Linux `wpa_supplicant` package [18] to implement the Secure Display Initialization EAP method (EAP-SDI). We also run a modified version of `hostapd` [19] in the RADIUS [20] server hosted as part of our cloud service. The user simply needs to press the power button on a new display and scan a QR code to both receive Internet access for the display and to securely connect it to the cloud service. For scanning the QR code and entering details such as the bootstrapping location, textual description and an image of the display with its surroundings, we provide the user with the mobile friendly web interface shown in Figure 4. This web interface is implemented with HTML5 and the `jQuery` mobile framework [21] to provide a mobile-friendly, responsive and rich interface. This web application is organized as a "wizard" in order to guide the user through the different phases involved in the configuration of the display. A `python` script on the display is responsible for searching the networks, running the EAP method in the background and starting the browser to show the QR code and the content.

A minimal one-time network configuration to create the long-term trust relation between the RADIUS server in the cloud and the local wireless infrastructure is needed. This could be done by the user or a professional. For instance, in our use-case scenario, the university IT department has taken care of this configuration once. Home network users and other small installations can perform the relatively simple configuration of the RADIUS functionality on their local servers or gateway devices by themselves. It is important to recall that establishing the initial trust relation is only a one-time configuration; it enables all future displays that join the network to be managed. However, this is not absolutely necessary and the user may also enter the SSID and password manually for every display to obtain network access. The user may also manually configure the "home page" of the browser engine in the display to point to the right cloud service and, finally, enter a username and password to authenticate this display for the access to the cloud service.

B. Management Interface

We currently use the `SQLite` [22] database server running in our cloud infrastructure to store all the information about the users' displays. This database is automatically populated with all the information made available by the bootstrapping phase. Thus, when a user installs and connects a new display with the cloud service, its unique identifier is automatically received and a new entry is created in the database. Optional information, such as the picture of the display and its location, are also added to the database when provided by the user. The management webpage is implemented with HTML5 and the `jQuery` mobile framework to allow users to access the management service from mobile devices. We use the `jQuery` short polling technique for updating the displays shown on the management interface.

The `python` script on the display that is responsible for searching for networks also regularly polls the cloud backend after successful connection to the Internet. This approach is necessary to check if the database has been updated with a new URL to be shown on the display. If the retrieved URL does not match the one currently being shown, the user intends to change the content. Thus, the `python` script re-directs the browser to the newly received URL. We do not perform a server push to keep the server-side implementation on the cloud-backend simple and save resources when monitoring database updates. Finally, we do not provide means for creating content playlists as users can already build them on existing services such as Youtube, Spotify, SlideShare, and several professional advertising playlist servers. Our management servers can integrate with these playlist services and other applications by simply pointing the content URL to them.

C. Remote Monitoring

Whenever a new display is authenticated to the cloud service, either by using the new EAP method or by manually entering the username and password, a shared secret is established between the display and the cloud [17]. Using this shared secret, we further derive two additional keys.

The first key is used to set up a Secure Shell (SSH) [23] tunnel from the display to the cloud. This is necessary as often the displays are located behind a Network Address Translator (NAT), therefore a connection in the reverse direction is not possible. This SSH tunnel provides security for the administrative access between the display and the cloud. The SSH tunnel is used for the lowest level of administrative shell access to the Linux-based display device, and it is intended to be used only for emergency remote software updates and reconfiguration. Most users will never need access on the shell level and may prefer to physically access the display device for major updates. However, when the number of displays grows and they are geographically distributed, it will become important to avoid physical access for even exceptional administrative processes. The tunnel also enables graphical monitoring of the displayed content, as explained next.

The second key is used to run a secure Virtual Network Connection (VNC) server on the display so as to allow low-level remote access and control for authenticated clients. In our implementation, we run the `x11vnc` [24] server on

the display. The SSH tunnel set up previously uses port forwarding to allow the cloud to connect to the `x11vnc` server on the display. In order to allow users logged into the management dashboard to have remote access to the displays directly from their browsers, we use the `noVNC` [25] browser-based VNC client. noVNC client uses standard HTML5 technologies such as WebSocket [26] and canvas, allowing it to be functional on most modern standard and mobile browsers such as Firefox, Chrome and Safari. As the `noVNC` client relies on WebSocket which is not natively supported by the `x11vnc` server, we run the `websockify` [27] WebSocket proxy in our cloud infrastructure. This proxy translates WebSocket connections coming from the noVNC client (at the user's browser) to regular socket traffic towards the x11vnc server on the display. Thus, the connection between the cloud and the display is protected over SSH while the connection between the cloud and the user's browser on any device is protected with HTTPS username and password. The management interface provides a unique URL link for every managed display. This unique URL already embeds the credentials for accessing the `x11vnc` server; the user can get remote access by simply clicking on the URL once logged in on the cloud-based signage management service.

V. Discussion

Even though our solution is cloud-based, this does not necessarily imply that the service has to be hosted in a public cloud accessed through the Internet. In many organizational contexts, security policies or simply a lack of trust in the cloud delivery model prevent such a deployment scenario. Indeed, a private cloud deployed within the enterprise network of the organization adopting the digital signage system would work as well. It is worth mentioning that the private cloud infrastructure within the premises of an enterprise does not need to be a full-fledged data center with powerful compute servers. In many situations, a rack or just a physical machine with virtualization capabilities are enough. Clearly, this deployment may limit the scalability of the service. However, it may be the only solution when the target deployment scenario requires accessing resources that are not otherwise available from the Internet. We have ourselves seen such constraints not only in sensitive environments such as hospitals and government agencies, but also in enterprises using legacy software such as room reservation systems. These constraints, however, have not prevented the adoption of our digital signage solutions.

As apparent from Figure 3, our current implementation of the management dashboard is minimalistic, without too many bells and whistles. When the number of displays is small, the information for all the displays being managed by a single user is directly accessible on one page. We do not believe in complex work flow management where content would be approved before being uploaded. With the explosion in the number of electronic displays, it would be expensive to assign employees or to hire external contractors to manage them. Our flexible signage management system adapts to the needs of the users, scaling both up and down. On the one hand, users are able to register to cloud service accounts to manage their displays themselves. On the other hand, organizations have the option to implement groups, access control policies, and delegation as the need for it arises. This would not only save costs but would also allow users to start managing their displays with little overhead and without having to initially rely on organizational processes.

One question that we are currently exploring is how to describe and register the display feature set and its relation to the installation context. Scaling non-interactive content to varying display dimensions can be done entirely in HTML5. However, it is necessary to also know contextual information such as how far the viewers are from the display and whether they actually can reach the touchscreen to fully exploit the screen resolution or interactive features.

Another challenge is to implement access control for displays. For entire displays, the access rights can be based on ownership, delegation, and a simple role-based model. However, when users share the same display and control different time slots or subareas, the access control becomes complicated [12]. Our current intuition is to leave such sub-display access control to the content application, but that can sometimes mean doubling the user accounts as well as the management effort.

VI. Conclusion

We have built a flexible management system for cloud-connected electronic displays such as digital signage. Motivated by an analysis of requirements collected from real digital signage deployments in enterprise settings, we have designed our management system by taking into account the entire lifecycle of a display. Our system is able to scale up from a few displays to several thousands. Our design philosophy has been governed by the minimum common requirements irrespective of the user and the deployment scenario. We have implemented this new management system as a cloud service and relied on modern web technologies so that it can be remotely accessed from a browser on any device.

We are currently investigating automated cloud-based mechanisms to reliably derive the operational state of the digital signs. We are also designing a role-based access control method that allows different actors to securely access the components of the systems they are authorized to interact with, according to their credentials.

Acknowledgments

This work was partially supported by TEKES as part of the Internet of Things program of DIGILE (the Finnish Strategic Center for Science, Technology and Innovation in the field of ICT and digital business).

REFERENCES

[1] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[2] J. Heikenfeld, P. Drzaic, J.-S. Yeo, and T. Koch, "Review paper: A critical review of the present and future prospects for electronic paper," *Journal of the Society for Information Display*, vol. 19, no. 2, 2011.

[3] A. Dementyev, J. Gummeson, D. Thrasher, A. Parks, D. Ganesan, J. R. Smith, and A. P. Sample, "Wirelessly powered bistable display tags," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '13, 2013, pp. 383–386.

[4] Y. Lu, S. Li, and H. Shen, "Virtualized screen: A third element for cloud–mobile convergence," *Multimedia, IEEE*, vol. 18, no. 2, pp. 4–11, February 2011.

[5] P. Simoens, F. De Turck, B. Dhoedt, and P. Demeester, "Remote display solutions for mobile cloud computing," *Computer*, vol. 44, no. 8, pp. 46–53, August 2011.

[6] "Ad cloud screens," http://onedigitalsolutions.com/cloud-digital-signage/, accessed: 2015-01-12.

[7] J. Case, M. Fedor, M. Schoffstall, and C. Davin, "A simple network management protocol (SNMP)," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 1157, December 1990. [Online]. Available: https://tools.ietf.org/html/rfc1157

[8] O. Storz, A. Friday, N. Davies, J. Finney, C. Sas, and J. G. Sheridan, "Public ubiquitous computing systems: Lessons from the e-campus display deployments," *Pervasive Computing, IEEE*, vol. 5, no. 3, pp. 40–47, 2006.

[9] K. Sugiura, M. Dayarathna, and A. Withana, "Design and implementation of distributed and scalable multimedia signage system," in *Ubiquitous and Future Networks (ICUFN), 2010 Second International Conference on*. IEEE, 2010, pp. 273–278.

[10] S. Clinch, M. Mikusz, M. Greis, N. Davies, and A. Friday, "Mercury: An application store for open display networks," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14, 2014, pp. 511–522.

[11] T. Linden, T. Heikkinen, T. Ojala, H. Kukka, and M. Jurmu, "Web-based framework for spatiotemporal screen real estate management of interactive public displays," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 1277–1280.

[12] K. Kuikkaniemi, V. Lehtinen, M. Nelimarkka, M. Vilkki, J. Ojala, and G. Jacucci, "Designing for presenters at public walk-up-and-use displays," in *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction*, ser. TEI '14, 2014, pp. 225–232.

[13] T. Heikkinen, T. Linden, T. Ojala, H. Kukka, M. Jurmu, and S. Hosio, "Lessons learned from the deployment and maintenance of ubi-hotspots," in *Multimedia and Ubiquitous Engineering (MUE), 2010 4th International Conference on*, August 2010, pp. 1–6.

[14] A. Friday, N. Davies, and C. Efstratiou, "Reflections on long-term experiments with public displays," *Computer*, vol. 45, no. 5, pp. 34–41, May 2012.

[15] "Broadsign: Digital signage software solutions," http://broadsign.com/, accessed: 2015-01-12.

[16] E. Oat, M. Di Francesco, and T. Aura, "MoCHA: Augmenting pervasive displays through mobile devices and web-based technologies," in *Proc. of the first IEEE Workshop on Developing Applications for Pervasive Display Networks (PD-Apps '14)*, March 2014, pp. 506–511.

[17] M. Sethi, E. Oat, M. Di Francesco, and T. Aura, "Secure bootstrapping of cloud-managed ubiquitous displays," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14. New York, NY, USA: ACM, 2014, pp. 739–750. [Online]. Available: http://doi.acm.org/10.1145/2632048.2632049

[18] "WPA Supplicant for Linux, BSD, Mac OS X, and Windows," http://hostap.epitest.fi/wpa_supplicant/.

[19] "hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator," http://hostap.epitest.fi/hostapd/.

[20] C. Rigney, A. C. Rubens, W. A. Simpson, and S. Willens, "Remote Authentication Dial In User Service (RADIUS)," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 2865, June 2000. [Online]. Available: http://tools.ietf.org/html/rfc2865

[21] "jQuery Mobile framework," http://jquerymobile.com/.

[22] M. Owens and G. Allen, *SQLite*. Springer, 2010.

[23] T. Ylonen, "Ssh–secure login connections over the internet," in *Proceedings of the 6th USENIX Security Symposium*, vol. 37, 1996.

[24] K. Runge, "x11vnc: a VNC server for real X displays," 2011, http://www.karlrunge.com/x11vnc/.

[25] J. Martin, "noVNC project website," 2011, https://github.com/kanaka/noVNC.

[26] I. Fette and A. Melnikov, "The WebSocket protocol," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 6455, December 2011. [Online]. Available: https://tools.ietf.org/html/rfc6455

[27] J. Martin, "websockify: WebSockets support for any application/server," 2012, https://github.com/kanaka/websockify.