

Cloudberry:

An HTML5 Cloud Phone Platform for Mobile Devices

Antero Taivalsaari and Kari Systä, Nokia

// One of the central benefits of a cloud phone is that nearly any of its customer-facing features can be changed on millions of devices all over the world almost instantly. //



THE SOFTWARE INDUSTRY is currently experiencing a paradigm shift toward Web-based software. Applications that used to be written for specific computer architectures, CPUs, operating systems, and devices are now increasingly written for the Web, to be executed inside a browser or a compatible runtime environment. In the future, most end-user software, including applications for both personal and business use, will be Web based. For the average computer user, the Web browser will be the primary software environment for most purposes, effectively displacing conventional operating systems from the central role that they had in the past.

We believe that the transition toward Web-based end-user software will have a significant effect not only on desktop computing but on mobile

devices as well. As of this writing, there's considerable momentum behind native, Web-connected mobile apps written for specific devices or operating systems; however, it's clear that Web-based software will also play a critical role in the mobile device space. In fact, we anticipate the battle of the decade to be between native apps and pure HTML5 Open Web applications.¹

In this article, we introduce Cloudberry, a novel HTML5-based cloud phone software platform developed at Nokia Research Center. One of the central benefits of a cloud phone is that nearly any customer-facing application or feature in such a device can be changed from the server side almost instantly to potentially millions of devices all over the world. This will shorten application and service deployments, update life cycles, and make it easier to

customize devices to different users and purposes. Another key benefit is multiple device ownership—that is, the ability to effortlessly access the same applications and data from different devices.

The Cloud Phone

A cloud phone is a mobile device in which all customer-facing functionality is downloaded and cached dynamically from the Web, including all the applications and even the entire top-level user interface (UI) of the device. For more information on mobile cloud systems, see the “Related Work in Mobile Cloud Systems” sidebar. A cloud phone will have several key characteristics.

User Experience Driven by the Web Runtime

In a conventional mobile device, the top-level user experience of the device is typically written natively; a Web browser is just one of the many applications provided by the system. In Cloudberry, the Web runtime is the centerpiece of the system that drives the entire top-level UI as well as all the applications. No native software is exposed to the user directly.

Applications Written as Web Applications

In Cloudberry, all mobile device applications—including core ones such as the phone dialer, contacts, calendar, messaging, music player, and maps—are written as Web applications.

Web-Based Application Development Model

The application development model for Cloudberry is based entirely on Web technologies—more specifically, on HTML, CSS, and JavaScript. There are no binary applications, and applications don't have to be compiled or linked statically; developers can use standard Web-based development tools.

RELATED WORK IN MOBILE CLOUD SYSTEMS

The Cloudberry system was inspired by our earlier experiences in building the Lively Kernel¹, one of the first truly interactive, exploratory JavaScript programming environments designed to run in a standard Web browser without any modifications or plug-ins.

In the mobile device space, the most similar system is the HP (Palm) WebOS.² The key differences between Cloudberry and WebOS relate to the application model. Whereas Cloudberry was built from the beginning around an open-ended HTML5 application model in which no explicit application installation is required, WebOS introduces a custom application model—built around the concept of stages and scenes—that requires explicit application packaging and installation. More recently, WebOS has started adopting an HTML5-style application model.

Other related systems include Google's Chrome OS (www.chromium.org/chromium-os) and Jolicloud (www.jolicloud.com). At this point, both Chrome OS and Jolicloud are targeted more toward desktop systems than mobile devices. However, it's reasonable to expect that these systems will eventually offer support for mobile devices, too.

In July 2011, Mozilla announced plans to build Boot to Gecko (B2G): a complete, standalone operating system for the open Web, with specific focus on mobile devices. At Mobile World Congress 2012, Mozilla announced that the Spanish telecommunications provider Telefónica intends to deliver B2G-based mobile devices. Mozilla also announced support for the project from Adobe and Qualcomm, and that Deutsche Telekom's Innovation Labs will join the project.

After Nokia discontinued the MeeGo platform development in 2011, Intel announced that the development of the MeeGo platform will continue in the Tizen effort and will use HTML5 as the primary application model (www.tizen.org). At this point, there's

no connection between Cloudberry and Tizen. From a technical perspective, a key difference is that Tizen uses the W3C Widget Application Model³—requiring explicit application installation—rather than the zero-installation HTML5 application model in which applications are effectively just links.

The term *cloud phone* has been used independently in a number of other contexts to date. Yan Lu and colleagues⁴ used this term to describe a thin mobile device that relies on network-streamed applications that run primarily on the server side, much like applications in Sun Microsystems' (now Oracle's) Sun Ray stateless client platform (www.oracle.com/us/technologies/virtualization/061984.html). The term also has been used for describing mobile telephone number swapping systems—intended primarily for developing countries—in which several users can share the same mobile phone. One of the leading commercial vendors in that space is Movirtu (www.movirtu.com).

As far as we know, Cloudberry is still the only system that has virtualized the entire end-user software environment of a mobile phone, so that all the applications of the mobile device (including the core applications) as well as the entire top-level user experience are downloaded from the cloud.

References

1. A. Taivalsaari et al., *Web Browser as an Application Platform: The Lively Kernel Experience*, tech. report TR-2008-175, Sun Microsystems Labs, 2008.
2. M. Allen, *Palm WebOS*, O'Reilly, 2009.
3. M. Cáceres, *Widget Packaging and XML Configuration Specification*, World Wide Web Consortium (W3C) recommendation, Sept. 2011; www.w3.org/TR/widgets.
4. Y. Lu, S. Li, and H. Shen, "Virtualized Screen: A Third Element for Cloud-Mobile Convergence," *IEEE Multimedia*, April–June 2011, pp. 4–11.

Web-Based Deployment Model and Transparent Updates

Cloudberry is a zero-installation platform—there's no notion of application installation in the conventional sense. Logically, each application is just like a webpage: it's just a link that the Web runtime will use to dynamically load and cache the necessary data and code for execution. Updates happen

automatically as the software changes on the server side. In fact, the entire suite of applications on each device can be entirely dynamic and changed (by the service provider or the users themselves) as frequently as necessary.

Master Copies in the Cloud

In a cloud phone, master copies of all applications and user data are stored in

the cloud. This lets the user switch between different devices easily because the same applications and data are accessible to all of his or her devices. Data is automatically backed up in the cloud as well as across all the user's devices.

Offline Use through HTML5

The ability to operate without an active network connection has become a

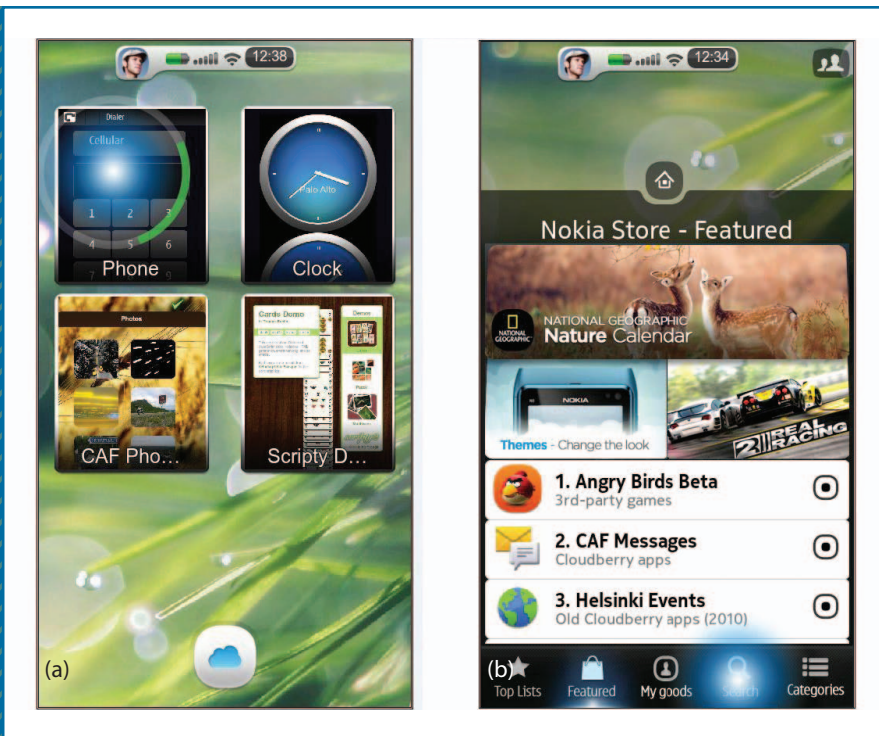


FIGURE 1. (a) Cloudberry 2011 desktop with four applications active but minimized on the desktop. (b) The application launcher/store. The store suggests three “featured applications.” In addition, the user can select applications from different categories or recommendation lists.

key element of any mobile phone. The Cloudberry system utilizes the HTML5 offline application support² as well as proprietary data-caching mechanisms to ensure that applications and their data are available when the device is offline. In general, because we use HTML5 for application execution, applications typically execute the majority of code on the client side, leveraging the mobile device’s CPU power and memory for application execution.

Application and User Interface Examples

Although a cloud phone’s technical architecture differs from that of a conventional mobile device, from the user’s perspective, a cloud phone doesn’t have to look or feel any different from a conventional device. In fact, one of the principal aims in our original cloud phone design was to make the

device look and feel instantly familiar to conventional mobile phone users. To reinforce this feeling, in our original Cloudberry 2010 system, we intentionally built all the applications to resemble the corresponding native applications on the Nokia N900 device. (We had the option to design things to be drastically different; Cloudberry has theming capabilities for different looks and feels far beyond those that are available on most mobile devices today.) In our more recent Cloudberry 2011 (from which we’ve taken most of the following examples), we leveraged cloud-specific capabilities much more extensively.

Figure 1 shows two screenshots from Cloudberry 2011; the UI style here used the notion of unlimited 2D space. Figure 1a shows a desktop with four open applications: phone, clock, a photo viewer, and a third-party HTML5

demo. Applications reside in an unlimited 2D space in which applications can float freely (which is difficult to convey in a static image). Visual effects such as a parallax effect reinforce the illusion of open, infinite space. By tapping any of the applications on the desktop, the user can bring an application to full-screen mode. Figure 2 shows some examples of full-screen applications.

In Figure 1b, the user has clicked on the cloud symbol on the bottom of the display to launch a new application. Instead of a conventional, static application launcher or grid, Cloudberry 2011 utilizes a dynamic, server-side-generated application store that consists of several featured applications, prearranged application categories, and top lists, as well as a list of those applications that the user has accessed recently. After the user chooses an application (which, in reality, is just a link), the Cloudberry client downloads and caches that application, or runs an existing version from cache if the application ran recently on the device. In the latter case, the system would also check if the previously cached application was recently updated on the server; if so, the system downloads the updated components transparently in the background.

Although our initial application set mimicked the native Nokia N900 applications, we recently built synthetic mashup applications that leverage resources available from the Web in various ways, so an application’s look and feel are unrelated to any existing native applications. Most third-party HTML5 applications also run in Cloudberry without modifications, although the UI layouts of such applications aren’t always optimal for the target device.

Using a Cloud Phone for the First Time

The general idea in a cloud phone is that applications and data can be used effortlessly on different devices. To use

the Cloudberry system on a specific device, the user must first enter credentials to establish a connection between the device and his or her identity in the cloud. After that, the top-level UI as well as existing open applications (if the user had any applications open on another device earlier) will become available on that device. User identification is performed only once per device. After the initial login, the device is subsequently associated with the specific user unless the user explicitly wants to detach his or her identity and purge all personal data from that device.

Technical Overview

From a technical perspective, the Cloudberry client stack consists of several components:

- a full-fledged WebKit-compatible Web browser (<http://trac.webkit.org/wiki/QtWebKit>) customized to support additional features;
- a set of downloadable top-level UIs that allow the device's entire look and feel and top-level user experience to change dynamically (until recently, all the top-level UIs in Cloudberry were written using the QML [QtQuick] language³ to maintain a clean separation between applications and the surrounding device UI, but we recently started implementing top-level UIs in HTML5 as well);
- a suite of Web applications that implement all the typical core applications found on a mobile phone (applications are built with adaptive UI layouts for flexible use on devices with different screen sizes, resolutions, and orientations; HTML5 supports offline use and automatic application updates, too);
- a data API that lets applications transparently store and automatically synchronize their data in a cloud datastore (the API's goal is to make data storage as seamless and



FIGURE 2. Screen snapshots of some full-screen core Cloudberry applications: contacts list, calendar, photo gallery, and map applications.

- transparent as possible; by default, all data is synchronized automatically in the cloud, however, data can be earmarked as local);
- a set of device APIs that let applications access device-specific functionality (such as initiating phone calls, sending text messages, reading GPS location information, and so on; device APIs are based on official W3C Device APIs [www.w3.org/2009/dap] wherever applicable, and proprietary APIs are used in those areas that standards don't yet cover);
- a domain-based, permission-based security model that restricts the use of device-specific functionality (such as device APIs) to only those applications from trusted domains; and
- an underlying process model to potentially run applications arriving from different domains in separate native OS processes.

Recent versions of the WebKit browser include similar capabilities ([http://trac.](http://trac.webkit.org/wiki/WebKit2)

[webkit.org/wiki/WebKit2](http://trac.webkit.org/wiki/WebKit2)), but our work in this area predates WebKit2 and provides more flexibility in choosing the split between applications, Web domains, and rendering processes.

Securely stored security-policy files determine the level of trust between a device and a specific Web domain. Applications must use permissions to explicitly request access to those features that are above and beyond the sandbox that the standard browser offers. This security model is an extension of the typical Web browser security model.⁵

Figure 3 shows a high-level architecture diagram of the Cloudberry system. The fact that it's built on top of a standards-compatible browser means that third-party HTML5 applications run without modifications. There's no hard split between core Cloudberry applications and third-party Web content, except in terms of access to the underlying device and platform capabilities.

Given that all the device's functionality is downloaded dynamically in Cloudberry, we placed special emphasis on security. Currently, we use several

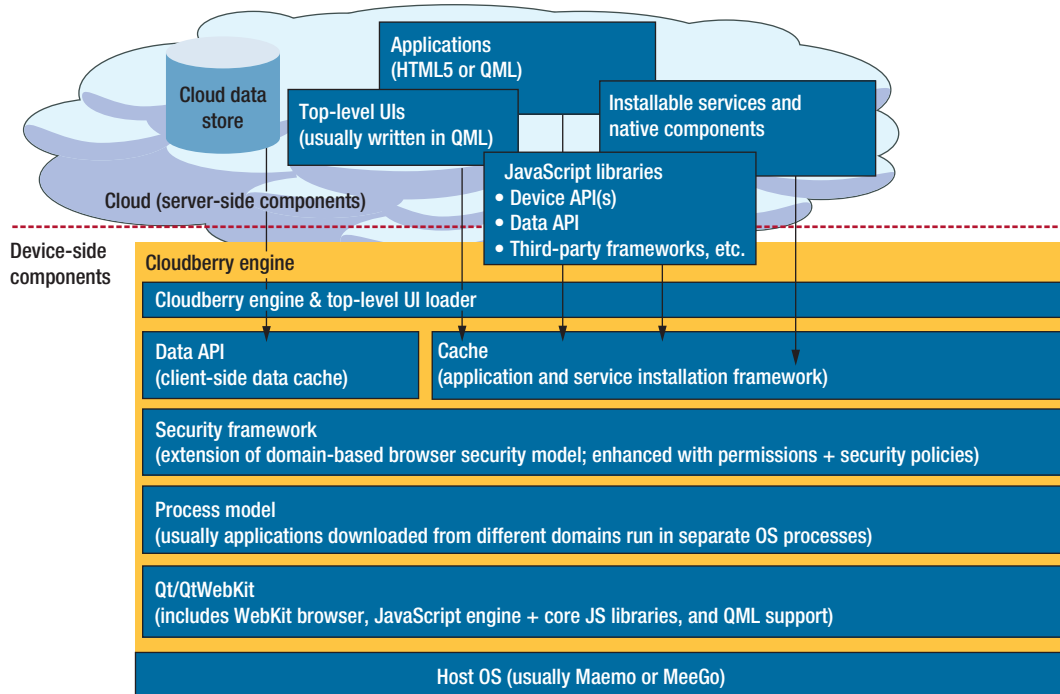


FIGURE 3. A high-level architecture diagram of the Cloudberry system. Nearly all the customer-facing features of the cloud phone are downloaded and cached dynamically from the Web.

proprietary extensions in this area, but we expect the rest of the industry to converge on these topics as the need for standardized security mechanisms for mobile Web applications becomes more widely apparent.

Evaluation and Discussion

On our original Nokia N900 target device, the Cloudberry system's performance could be described as adequate at best. Over the course of the project, the performance of mobile devices and their Web runtimes has improved markedly, particularly with newer devices, improving user experience considerably. Dramatic improvements in JavaScript virtual machine performance have also helped us. In general, we're rapidly reaching a point in which Web engine performance will no longer be a barrier for building a mobile

device entirely around a Web-based user experience.

Architecture Validation

One misconception that we frequently run into when discussing the cloud phone concept is related to client devices' "thinness." Historically, cloud computing has implied the use of thin clients—that is, an approach in which the majority of computation is performed on the server side. Consequently, upon first hearing about the Cloudberry system, most people assume an execution model in which the majority of execution takes place in the cloud.

In practice, any system that relies on the HTML5 application model, especially on its offline execution capabilities, isn't really a thin client at all. Although the Web environment offers flexible mechanisms for balancing the

computation needs between the client and the server through libraries such as Node.js (<http://nodejs.org>), in reality, the majority of execution in HTML5-based environment systems takes place on the client. The execution model that the Cloudberry system uses is actually surprisingly similar to other mobile software platforms that rely on a sandboxed virtual machine environment (for example, Java ME⁶ or Android). From this perspective, the Cloudberry system's overall architecture doesn't need any particular validation.

A key difference with Cloudberry compared to systems such as JavaME or Android is its ability to update applications and their components at a dramatically faster pace by leveraging the fact that any changes made to the applications' master copies and their components on the server side are

reflected almost immediately to potentially millions of client devices across the world. We haven't yet tested the scalability of these features, but several topics related to the system's security and overall scalability are currently under investigation.

Remaining Issues and Challenges

We discussed the limitations of the Web browser as a software platform extensively in earlier work.^{7,8} Most of these observations apply to the Cloudberry system as well. If we had to pick a single issue hampering the wide-scale deployment of an HTML5-based mobile platform, it would be the lack of standardization. Although HTML5 and related W3C standard activities play a critical role in turning the Web into a compelling application platform, the current feature set offered by an HTML5-compliant Web browser is still incomplete for real-world applications. For instance, the device APIs that are under definition in W3C (www.w3.org/2009/dap) offer only limited access to platform features available in personal computers and mobile devices today. Proprietary APIs are still necessary to access device and platform features that are above and beyond the capabilities offered by standards. This considerably reduces the portability of applications and the overall value of the cloud phone proposition.

We predict that another major round of standardization will be necessary to establish a more complete Web application platform beyond HTML5. A critical goal in that standardization activity will be to more comprehensively virtualize the underlying operating system and device capabilities, as well as to ensure that the necessary security mechanisms are in place to securely access all the platform capabilities. Incidentally, the World Wide Web Consortium has started work on HTML.next (www.w3.org/wiki/HTML/next).


ABOUT THE AUTHORS



ANTERO TAIVALSAARI is a distinguished engineer at Nokia. His research interests include Web application technologies and Web-based software development especially for mobile devices. Taivalsaari has a PhD in computer science from the University of Jyväskylä, Finland. Contact him at antero.taivalsaari@nokia.com



KARI SYSTÄ is a professor of computer science at Tampere University of Technology, Finland. His research interests include the Java platform and micro edition. Systä has a PhD in computer science from Tampere University of Technology. Contact him at kari.systa@tut.fi.

At this point, Cloudberry is a research system with no immediate product plans. However, many of the concepts explored in the project are already finding their way to commercial products. We believe that the trend toward Web-based software will cause a paradigm shift in the software industry from conventional binary applications to dynamically delivered Web applications. In the future, the use of conventional binary programs will be limited mainly to system software, whereas the vast majority of end-user software will be developed using Web technologies. The Cloudberry system, for its part, has proven that such a transition is feasible in the mobile device space. 

References

1. T. Mikkonen and A. Taivalsaari, "Apps vs. Open Web: The Battle of the Decade," *Proc. 2nd Workshop Software Eng. for Mobile Application Development* (MSE 11), 2011; www.mobileseworkshop.org/papers/6-Mikkonen_Taivalsaari.pdf.

2. I. Hickson, "HTML5: A Vocabulary and Associated APIs for HTML and XHTML," World Wide Web Consortium (W3C) Internet draft, work in progress, 2012.
3. R. Rischpater and D. Zucker, *Beginning Nokia Apps Development: Qt and HTML5 for Symbian and MeeGo*, APress, 2010.
5. M. Zalewski, *Browser Security Handbook*, Google, 2008; <http://code.google.com/p/browsersec/wiki/Main>.
6. R. Riggs et al., *Programming Wireless Devices with the Java 2 Platform*, 2nd ed., Addison-Wesley, 2003.
7. A. Taivalsaari et al., *Web Browser as an Application Platform: The Lively Kernel Experience*, tech. report TR-2008-175, Sun Microsystems Labs, 2008.
8. A. Taivalsaari and T. Mikkonen, "The Web as an Application Platform: The Saga Continues," *Proc. 37th Euromicro Conf. Software Engineering and Advanced Applications* (SEAA 11), IEEE CS, 2011, pp. 170–174.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.