International Journal of Computer Applications Technology and Research
Volume 4– Issue 9, 655 - 663, 2015, ISSN: 2319–8656

WAP, HTTP and HTML5 Web Socket Architecture Analysis in Contemporary Mobile App Development

Eke B. O.
Department of Computer Science, University of Port Harcourt
Port Harcourt, Nigeria

Onuodu F. E.
Department of Computer Science, University of Port Harcourt
Port Harcourt, Nigeria

Abstract: Accessing current and accurate information anywhere and at anytime is becoming a growing interest nowadays. Wireless Application Protocol (WAP) is an application protocol that creates an opportunity to access information of any interest from WAP servers using mobile phones. WAP is an enabling technology based on the Internet client server architecture model, for developing client application for handheld devices or other wireless terminal which usually have less powerful CPU's, less memory, very restricted power consumption, smaller and variant displays, phone keypads etc. This paper analyses the features of WAP in relation to the well established HyperText Transfer Protocol (HTTP) technology, the web socket API innovations introduced in HTML5, the recent improvements in mobile devices processing capacity by connecting to cloud services and how application can be developed on them using modern tools. The features that are more adapted to client development of micro-devices are used for the technology application test.

Keywords: WAP, HTTP, HTML5, Web Socket API, Mobile App, Client, Servers, Cloud

1. INTRODUCTION

Recently, protocol technologies that enable handheld wireless devices to retrieve information have increased, presenting a more constrain on the computing environment compared to desktop computers. Some of the major protocols used in the wireless devices development include Wireless Application Protocol (WAP) and the HyperText Transfer Protocol (HTTP). The recent introduction of HTML5 and its associated Web Socket API has improved developers experience in building mobile applications and even connecting seamlessly to the cloud. services. WAP is an open protocol for wireless messaging, it provides the same technology to all vendors regardless of the network system. The WAP standard described a protocol suite allowing the interoperability of WAP equipment and software with different network technologies, such as GSM and IS-95 (also known as CDMA). The Hypertext Transfer Protocol (HTTP) on the other hand is an application protocol for distributed, collaborative, hypermedia information systems.

The standard development of HTTP was coordinated by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs), most notably RFC2616 [1], which defines HTTP/1.1, the version of HTTP in common use.

HTTP functions as a request-response protocol in the client-server computing model. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

In this paper the features of these two protocols that are more adapted to client development of micro-devices client environments that are reusable and user-friendly are presented.

1.1 WAP Technology

The WAP platform is an open specification that addresses wireless network characteristics by adapting existing network technologies (and introducing new ones where appropriate) to the special requirements of handheld wireless devices. WAP is made up of WAP Session Protocol (WSP), WAP Transaction Protocol (WTP), Wireless Transport Layer Security (WTLS),WAP Datagram Protocol and the Bearer that support WAP architecture and the internet layer on the other hand connected by the WAP Gateway [2]. WAP intends to standardize the way wireless devices (mobile phones, PDA, and so forth) access Internet data and services. WAP's reuse of existing Internet protocols eases the development of WAP services for Wireless Markup Language (WML) and WMLScript. Wireless handheld devices preset a more constrained computing environment and platforms, compared to desktop computers which most of the Internet technology was developed for. The handheld devices tend to have less powerful CPU's, less memory, very restricted power consumption, smaller and variant displays, phone keypads etc.

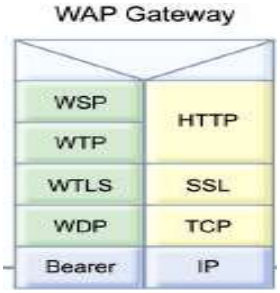

Figure 1: WAP technology GateWay

Furthermore, the wireless networks present additional constraints as communication infrastructures. They have less bandwidth, more latency and less connection stability and unpredictable availability. WAP intends to overcome these difficulties by being interoperable, have scaleable quality of service, efficient in the mobile network resources, reliable and secure.

WAP allows carriers to strengthen their service offerings by providing subscribers with the information they want and need while on the move. Infrastructure vendors will deliver the supporting network equipment. Application developers and content providers delivering the value added services are contributing to the WAP specification. Enabling information access from handheld devices requires a deep understanding of both technical and market issues that are unique to the wireless environment. The WAP specification was developed by the industry's best minds to address these issues[2]. Because, WAP is a standardized way for delivering Internet data over wireless networks and capable of addressing the unique characteristics of mobile terminals and wireless networks.

2. THE WAP ARCHITECTURE

The WAP standard defines two essential elements: an end-to-end application protocol and an application environment based on a browser. The application protocol is a communication protocol stack that is embedded in each WAP-enabled wireless device (also known as the user agent). The server side implements the other end of the protocol, which is capable of communicating with any WAP client. The server side is known as a WAP gateway and routes requests from the client to an HTTP (Hyper Text Transfer Protocol) (or Web) server. The WAP gateway can be located either in an Operator premises as illustrated in Figure 1. or in WAP application provider premises with the web server as shown in Figure 2.

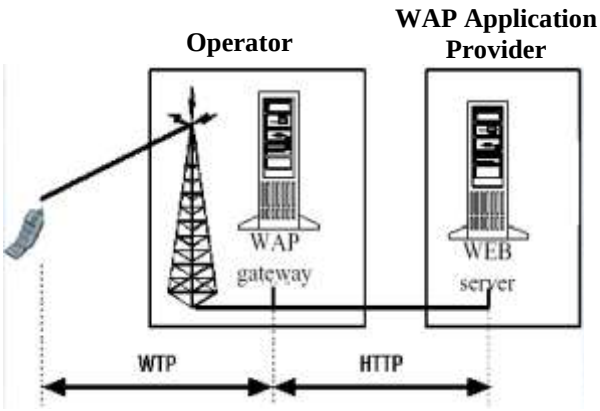

Figure 1. Gateway equipment in the operator premises

Irrespective of the WAP architecture used in an implementation, the core issues is the shielding of the micro-devices with less powerful CPU's, less memory, very restricted power consumption, smaller and variant displays etc from the challenging Web client processing and handling capacity.

In the WAP network the client communicates with the WAP gateway in the wireless network. The WAP gateway translates WAP requests to WWW requests, so the WAP client is able to submit requests to the Web server. Also, the WAP gateway translates Web responses into WAP responses or a format understood by the WAP client [3].

The wireless application environment provides WAP micro browser for interaction between WAP (web applications) and wireless devices. This browser relies on WAP markup languages such as WML (Wireless Markup Language), WML Script and XHTML MP (Extensible Hypertext Markup Language Mobile Profile.

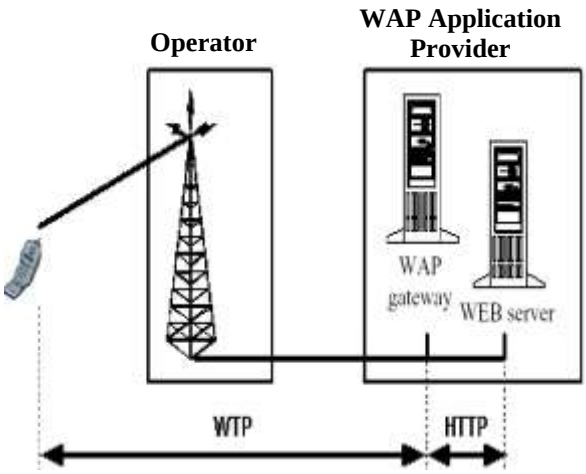

Figure 2. WAP Gateway in the WAP application provider premises

2.1 The WAP Programming Model

The WAP programming model is similar to the Web programming model with matching extensions, but it accommodates the characteristics of the wireless environment. The WAP programming model is based heavily on the Web programming model. But how does the WAP gateway work with HTML (Hyper Text Markup Language)? In some cases, the data services or content located on the Web server is HTML-based.

Some WAP gateways could be made to convert HTML pages into a format that can be displayed on wireless devices [3]. Because HTML wasn't really designed for small screens, the WAP protocol defines its own markup language WML, WML Script, and XHTML MP is the languages that are specifically designed to develop WAP applications for Mobile devices. These languages adhere to the XML standard and are designed to enable powerful applications within the constraints of handheld devices. In most cases, the actual application or other content located on the Web server will be native WAP contents created with WML (XHTML MP) or generated dynamically using WML Script, Java Servlets or JSP (Java Server Page), or other server side programming languages.

WML is an XML-based markup language that was designed especially to present WAP content on a wireless terminal. WML can preserve the content of variables between different WML pages. The basic unit of WML is the card that specifies a single interaction between the user and the user agent. Multiple cards are grouped together in decks, which is the top most element of a WML file. When the user agent receives a deck, it activates only the first card in the deck. There are no functions to check the validity of user input or to generate messages and dialog boxes locally using WML. Therefore, to overcome this limitation, WML Script was developed. WML Script, which is based on ECMA Script (the standard for java script), is a language that can be used to provide programmed functionality to WAP applications. It was defined to enable

the execution of scripts on WAP devices. The goal of using WML Script is to reduce the number of turn around between the client and the server. It is part of the WAP specification, and it can be used to add script support to the client. Its difference from ECMA Script is that it is compiled into byte code before it is sent to the client. The main reason for this is to cope up with the narrowband communication channels and to keep client memory requirements to a minimum. XHTML is a markup language used to create richer web content on an ever-increasing range of platforms including mobile handsets. It is similar with HTML in its tag definition and syntax, but it adds modularity and enforces strict adherence to language rules. It brings a clear structure to web pages, which is especially important for the small screens and limited power of mobile devices. This WAP programming Model is illustrated in figure 3. The XHTML MP is a mobile adaptation of XHTML by excluding those features not appropriate for devices with small screens. It is a strict subset of XHTML that includes additional elements and attributes that are useful in mobile browsers with additional presentation elements and support for internal style sheets.

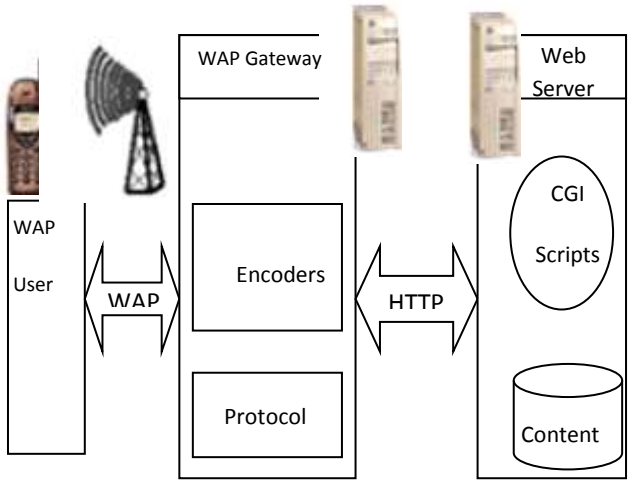

Figure 3. The WAP Programming Model

Mobile browsing technology is evolving from WAP 1.x to WAP 2.0, by introducing different enhancements for mobile content development. Especially WAP 2.0 provides support for protocols such as IP, TCP and HTTP. This provides interoperable optimizations suitable to the wireless environment and to the environment that permits wireless devices to utilize existing Internet technologies. WAP 2.0 also provides different application environment, which enables delivery of information and interactive services to wireless devices.

WAP standard defines the future of wireless browsing technology based on the WML, XHTML MP and WAP CSS (WAP Cascading Style Sheet). Both WML and XHTML MP are a reformulation of the XML. XML is a language for marking up structures in text documents and supports the UTF-8 (8 bit Unicode Transformation Format) coding standard. The UTF-8 coding standard supports several languages character set. So WML and XHTML MP can be used to create WAP pages that are encoded as UTF-8. Browsing from wireless terminals supporting UTF-8 encoding becomes possible.

2.3 HTTP Architecture

The innovations that Berners-Lee added to the Internet to create the World Wide Web had two fundamental dimensions: connectivity and interface [4]. He invented a new protocol for the computers to speak as they exchanged hypermedia documents.

HTTP stands for **Hypertext Transfer Protocol**. It is a TCP/IP based communication protocol which is used to deliver virtually all files and other data, collectively called resources, on the World Wide Web. These resources could be HTML files, image files, query results, or anything else.

A browser works as an HTTP client because it sends requests to an HTTP server which is called Web server. The Web Server then sends responses back to the client. The standard and default port for HTTP servers to listen on is 80 but it can be changed to any other port like 8080 etc.

There are three important things about HTTP, which should be noted:

- **HTTP is connectionless:** After a request is made, the client disconnects from the server and waits for a response. The server must re-establish the connection after it processes the request.
- **HTTP is media independent:** Any type of data can be sent by HTTP as long as both the client and server know how to handle the data content. How content is handled is determined by the MIME specification.
- **HTTP is stateless:** This is a direct result of HTTP's being connectionless. The server and client are aware of each other only during a request. Afterwards, each forgets the other. For this reason neither the client nor the browser can retain information between different requests across the web pages.

Figure 4 below shows where HTTP Protocol fits in communication:

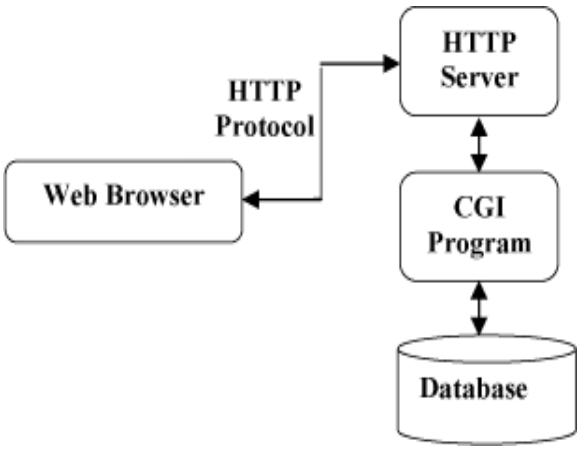

Figure 4. The HTTP Protocol in Communication Model

The set of common methods for HTTP/1.0 is defined below. Although this set can be expanded.

The model makes it possible for a client to reach services on a large number of origin servers; each addressed by a unique Uniform Resource Locator (URL). The content stored on the

servers is of various formats, but HTML is the predominant. HTML provides the content developer with a means to describe the appearance of a service in a flat document structure. If more advanced features like procedural logic are needed, then scripting languages such as JavaScript or VB Script may be utilised.

Figure 5 below shows how a WWW client request a resource stored on a web server. On the Internet, standard

communication protocols, like HTTP and Transmission Control Protocol/Internet Protocol (TCP/IP) are used.

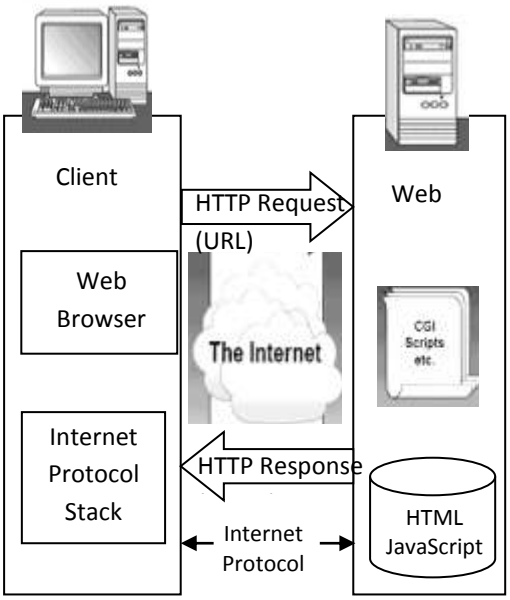

Figure 5. A HTTP Client-Server Request-Response Architecture

3. ANALYSIS OF CONTEMPORARY TECHNOLOGIES

It is clear that what we are considering in this work is web mobile development technologies supported by WAP and HTTP. These protocols have a way of supporting web development on mobile devices and even a native-web application development via devices internal client scripting languages. These use the operating systems browser as a runtime environment and are runnable on all mobile operating systems without any installation process [5].

The programming languages to create web applications on WAP include WML,WMLS but on HTTP, HTML5, CSS and JavaScript are used. Those allow platform independent development. The application can be opened by accessing a specific website.. A specifically developed and adjusted website mocking a native application in design and functionality [6]. With particular techniques, these web apps can also be used without an Internet connection, as long as the user has saved them on the device once and not cleared the browser's cache. In iOS, this is achieved via the function "add to home screen". A third way of developing mobile applications is to combine native and web development. Connecting a container application written in platform specific code with a web app containing the application logic and the user interface leads to a program referred to as "hybrid". In the most extreme case, the native app contains only a single browser window in full screen, running the web app [7]. The corresponding element is called UIWebView in iOS and WebView in Android and Windows.

As the different development approaches for mobile applications have been shown, the question remains which of these – depending on the applied criterion – offers advantages and disadvantages. The criteria "Content vs. Experience", "Performance", "APIs", "Distribution" and "Development Cost" are to be considered. Those are only a subset of possible criteria and represent those with a clear difference in results.

3.1 Analysis of HTML5, HTTP and Mobile Device Development

Before the development of HTML 4 in 1997, World Wide Web Consortium (W3C) believed that XML is the future of the web even with the good features of HTML4, this view never changed. The strict definitional guideline for development of XML and need to abide by the rules of coding made it look exiting. The development of XHTML lend credence to this view but with the release of HTML5 to simplify developing for the web many new elements were added, along with the addition of several new JavaScript API's. Previous to HTML 5, playing media on a browser required a plug-in or an application installation [6]. For example playing a Flash game requires Adobe shockwave player but now thanks to HTML5 and its new multimedia elements audio, video and canvas that could be a thing of the past. Although HTML5 is not without faults most see it as a huge leap forward for web development. Its release plays a vital role how web games are developed. Recently there has been a massive expansion of casual 'pick up and play' games which are frequently played on social networking sites, smart phones, tablets and web sites this has led to a major shift in the game development industry.

The Canvas element provides a way for developers to draw and manipulate 2D images using HTML and JavaScript to implement movement of a canvas element the image must be deleted and redrawn continuously by the web OS. With the canvas element 2D games can be easily implemented and with the use of vector based images they can be manipulated without losing any quality. Websites such as FaceBook have already begun providing an abundance of games showing the effectiveness of HTML5 canvas. The WebGL API is another exciting and innovative feature that allows the customisation of 3D objects by allowing JavaScript to communication with the users GPU. WebGL provides an API that allows 3D graphics to be used with canvas. This could potentially make web browsers a valid gaming platform, which would revolutionise game development.

WebSocket technology provides full-duplex bi-directional communication channels over a single TCP socket in both web browsers and web servers [8]. The WebSocket API provides features in JavaScript that allow the implementation of real-time interactions on applications. For example web based games will be able to provide instant interactions with game objects (such as the canvas images) and provide in game chat functionalities. Combining Canvas and WebSockets allows for the development of a multiplayer web browser

based games framework [9]. The effectiveness of these new HTML5 features provide information on how these technologies can and are being used.

Normal web communication is achieved using HTTP, the problem with this is that it only allows transfer to occur in one direction at a time. HTTP communications also requires constant web page requests each time new data is required. This results in slow communication between client and server. Current technologies such as Ajax and Comet attempt to speed up this transfer. These technologies attempt to simulate a full-duplex connection, but they are merely a hack and still truly only provide one way communication. Comet uses methods such as long-polling or streaming, these methods however still involve using HTTP requests which causes latency issues. This is where WebSockets come in; HTTP was not designed to support real time communication but WebSockets upgrade this HTTP protocol to a WebSocket protocol. This enables true full-duplex, bidirectional connections, what this signifies is the ability to create real time multiplayer games on web applications. WebSockets portray a massive advancement for real time applications on the web. Although WebSockets do not render Ajax completely obsolete they do however supersede its solution for real time functionality. Originally WebSockets were implemented into all web browsers providing developers with the means to begin creating interactive real time games, however due to security vulnerabilities WebSockets were disabled in a number of browsers including Mozilla Firefox 4 [10].

This security issue has been corrected in the latest version of the WebSocket protocol, which means we can still expect to see multiplayer web based games coming becoming available in the latest web browser OS versions. The potential for WebSockets when used alongside other HTML5 technologies such as Canvas and WebGL is incredible, a number of games have already been created displaying different creative features built with WebSockets. Figure 3 shows an online multiplayer game "Rawkets" which looks similar to asteroids only offering multiplayer free for all game play. The game is in fact quite enjoyable and shows the potential of games, signifying that real time multiplayer Games could begin to expand all over social networking sites in the near future. Another game which adds to this theory is the online social chatting game (Figure 4) "Rumpetroll", which transforms players into a tadpole floating around space. The real time chat feature from WebSockets can clearly be seen here as the messages sent between players reach each other almost instantaneously. Using WebSockets in these games shows the availability of real time collision detection and chat features, with almost no latency being displayed WebSockets really is the "*Quantum Leap"* [11] that will bring real-time multiplayer games to the web and mobile devices. The Pusher API [10] enables developers to easily add WebSocket functionality to web games and mobile games, they also offer a number of libraries to be used within your game that explain how to effectively use Pushers client libraries.

With all the new elements of HTML5 and the new JavaScript features there comes an abundance of third party software to optimise features and lessen the learning curve required to delve into these innovative elements.

3.2 HTML5 and the Cloud

It is clear that HTML5 has cross-platform capability and its support is available in any latest mobile device's browser. So, application made in HTML5 can be run on any devices. Ability to use same application and data from different devices without tedious installations. Here applications and data are stored on cloud and not bound to particular devices. So, we can use it on anywhere, anytime and on any device.HTML5 is lightweight than other alternative like Flash, etc. It has not required extra plug-ins because it has built in support for video, audio, canvas, etc.

If WebSocket connection mechanism is used with deflate compression mechanism in HTML5, network consumption will become efficient which supports HTML5-based cloud phone software platform concept. In that, HTML5 worked as software platform for mobile devices in which all end user functionality of devices is downloaded and cached dynamically from web including all applications [10].

The offline application capability and web storage of HTML5 enabled mobile web browser to bring an offline user interface to users. Web storage is a feature that is intended to overcome the limitation of HTTP cookies. Despite it's mainly purpose is for state management mechanism, cookies can be used for storing information at client side, at limited size. Using Web storage, client side will have enough space for storing more data. In order to making it capable to operate while offline, Offline web application feature is used. Offline web application allows set of HTTP objects is able to be accessed from browser without having to connect to Internet.

If mobile application is mainly used to display and interact with online content or services, it is better to avoid the native choice. However, if mobile application is mainly used offline, a native app will offer a better user experience.

WebGL[9] APIs have only recently become available in major browsers. WebGL will provide support for highperformance, direct manipulation 3D graphics content that can run without installation in major web browsers. We believe that WebGL will allow even high-performance gaming applications to run efficiently in a standard web browser.

3.3 HTML5 Mobile Application Cloud Architecture

In mobile application cloud architecture HTML5 on the mobile browser connects to the internet which in turn connects to a cloud server via a firewall. The backend data and the cloud services are connected to the cloud server which then provides the services to the mobile devices. This operation is supported by the mobile device OS and the HTML5 components that supports the cloud services.

Common characteristics of mobile web development frameworks supported by this architectures include [12]:

Cross-platform*:* Support for multiple mobile device Platforms allows to deliver app to a wide range of users.

Lightweight*:* Due to current bandwidth limitations, a stronger emphasis on lowering file weight is placed into mobile web development frameworks.

Optimized for touch screen devices*:* Fingers as input devices instead of mouse cursors provide an extra set of challenges in user interface design. Mobile web development frameworks

provide standard user interface elements and event-handling specifically for mobile device platforms.

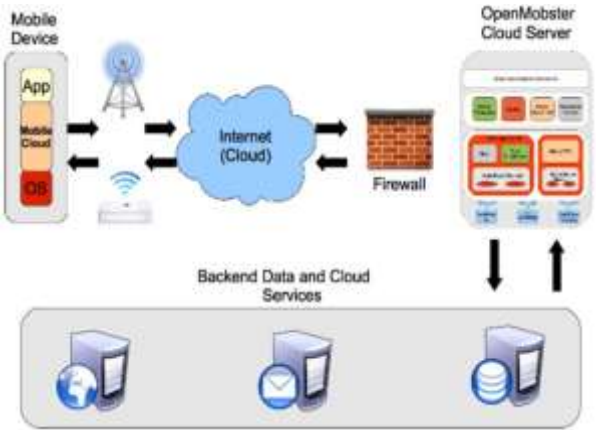

Figure.6. The open mobster architecture [12]

Uses HTML5 and CSS3 standards*:* Most mainstream mobile devices have web browsers that support HTML5 and CSS3, and so mobile web development frameworks take advantage of new features available in these upcoming W3C specifications for a better user experience.

3.3.1. Use of HTML5 Mobile Cloud Computing

The HTML5 breakthrough is mainly represented by the extreme simplification of the web content creation and by the generality of browsers, alleviated from the needs of complex plug-ins.

These new features are ensured by completely new syntactic elements (like <video>, <audio>, <canvas>), by hooks toward other standards (*e.g.* the possibility of using Web Socket [8]) or by extending existing elements for recent interaction modes (*e.g.* from on Click to on Touch functionalities).

The <canvas>[9] element is the enabler for real-time drawing of complex graphical content (paths, boxes, circles, characters and images) and gives the possibility for dynamic update and creation of the web content on the fly, by using JavaScript [8].

Beyond AJAX (Asynchronous JavaScript And XML), in order to ensure content delivery, HTML5 uses Web Socket, a new mechanism for establishing a connection between the server and the client.

4. DESIGN AND TECHNICAL CHALLENGES OF WAP AND HTTP

There are design and technical challenges that these technologies face which must be remedied in the new technologies. When new technologies such as HTML5 makes entry into mobile application development, there is a need to correct and improve the challenges. These design and technical challenges are found in HTTP and WAP.

4.1 HTTP Design Challenges

The major challenge of HTTP is in its very nature which is its Request-Response design as illustrated in figure 5. In the time HTTP was developed it was design to give a single response for every single request. This means that for 1 million response there is a corresponding 1 million request. This approach of having HTTP Response for each HTTP Request, is a huge drawback when having dynamic real-time updates sent from the server to the client (browser), requiring a lot of extra-traffic for each packet. This is too much burden on the network as well as on the server. Mobile lightweight nature low network capability will not easily adapt to that challenge. This made designers to develop Asynchronous JavaScript And XML (AJAX) in the preceding years when XML was seen as the answer even by W3C.

However, AJAX was not a complete solution since it has certain drawbacks inherent in HTTP. On the one hand, AJAX (Asynchronous JavaScript And XML) is used for creating asynchronous request from the client to the server and for receiving server side responses in the same manner. Hence, any server request results in creating an XMLHttpRequest. Once the server receives this request, it parses it and sends the corresponding response to the browser as HTTP Response. This improves the dynamic real-time updates sent from the server to the client but does not eliminate the HTTP circle.

On the other hand, HTML5 Web Socket API supports bidirectional, full-duplex communication over a single socket. A Web Socket based communication is established by sending from the client to the server, a simple HTTP Request for upgrading its connection type to Web Socket. If the server positively answers with the Web Socket upgrade message, the subsequent messages are exchanged by using the Web Socket API. The compression mechanisms vary with the connection type: The HTTP provides both gzip and deflate supports [8], while in the Web Socket case, deflate is the only available mechanism [8]. Considering the user interaction, HTML5 reconsiders and extends the previous elements so as to deal with the interaction modes.

Hence, mobile application developed with HTML5 framework will give better performance over web based mobile cloud application requiring high bandwidth. Mobile application will also run across platform efficiently and it will not have any constraint of any proprietary system. Resource starved application which have required more resources like memory, processing will easily run on mobile devices by using appropriate cloud services.

4.2 WAP Design Challenges

We have clearly specified that the languages supporting WAP technology is WML and WMLS(a subset of ECMAScript). One of the major set back of the technology is the language-WML. WML cut users off from the conventional HTML Web, leaving only native WAP content and Web-to-WAP proxi-content available to WAP users. The technology was provided custom-designed content by reducing complexity interface as the citizens of many nations are not connected to the web at the inception of the technology and have to use government funded and controlled portals to WAP and similar non-complex services.

There was equally an under-specification of terminal requirements which meant that compliant devices would not necessarily interoperate properly. This resulted in great variability in the actual behavior of phones, principally because WAP-service implementers and mobile-phone manufacturers may have not obtained a copy of the standards or the correct hardware and the standard software modules. As an example, some phone models would not accept a page more than 1 Kb in size; others would downright crash. The user interface of devices was also underspecified: as an

example, accesskeys (e.g., the ability to press '4' to access directly the fourth link in a list) were variously implemented depending on phone models (sometimes with the accesskey number automatically displayed by the browser next to the link, sometimes without it, and sometimes accesskeys were not implemented at all).

Mover WML have Constrained user interface capabilities making terminals with small black-and-white screens and few buttons, like the early WAP terminals, face difficulties in presenting a lot of information to their user, which compounded the other problems: one would have had to be extra careful in designing the user interface on such a resource-constrained device which was the real concept of WAP.

Developers were also not offered good authoring tools that could allow content providers to easily publish content that would interoperate flawlessly with many models, adapting the pages presented to the User-Agent type. This could have provided solution to the interface constrain WML problems. However, the development kits which existed did not provide such a general capability. Developing for the web was easy: with a text editor and a web browser, anybody could get started, thanks also to the forgiving nature of most desktop browser rendering engines. By contrast, the stringent requirements of the WML specifications, the variability in terminals, and the demands of testing on various wireless terminals, along with the lack of widely available desktop authoring and emulation tools, considerably lengthened the time required to complete most projects. The many mobile devices supporting XHTML, and programs such as Adobe Go Live and Dreamweaver offering improved web-authoring tools, it is becoming easier to create content, accessible by many new devices. This created massive exodus from WML.

Lack of user agent profiling tools and database of device capabilities made it difficult to know which devices users are using to get the correct content.

Often when a technology neglects content providers then it seem to be heading to extinction. Some wireless carriers had assumed a "build it and they will come" strategy, meaning that they would just provide the transport of data as well as the terminals, and then wait for content providers to publish their services on the Internet and make their investment in WAP useful. However, content providers received little help or incentive to go through the complicated route of development. Others, notably in Japan, had a more thorough dialogue with their content-provider community, which was then replicated in modern, more successful WAP services such as i-mode in Japan or the Gallery service in France.

The challenge of lack of openness in the technology also scared away developers. Many wireless carriers sold their WAP services as "open", in that they allowed users to reach any service expressed in WML and published on the Internet. However, they also made sure that the first page that clients accessed was their own "wireless portal", which they controlled very closely. Some carriers also turned off editing or accessing the address bar in the device's browser. To facilitate users wanting to go off deck, an address bar on a form on a page linked off the hard coded home page was provided. It makes it easier for carriers to implement filtering of off deck WML sites by URLs or to disable the address bar in the future if the carrier decides to switch all users to a walled garden model. The vendors, phone manufacturers and W3C must have learned the importance of developers in the sustenance of any given technology.

5. PERFORMANCE ANALYSIS OF WAP, HTTP AND HTMLS

The performance of the technologies on varying phones and mobile devices will be analyzed in other to find the adaptability of the technologies to mobile developers and how widely applications in the market places have deployed the technologies. The period that will be under consideration is 2005-2015 a period of ten years and the data that will be used in the analysis is a secondary data. The analysis will use WML adaptability and development for evaluating WAP technology. HTML version 3 and below for simple HTTP request-responses and HMTL5 for web socket and other recent improvements [13].

In figure 7 the plot of the performance of this technologies estimated based on the number of deployments, resulting in mobile application development over a period of ten years gathered from W3C and Mobile vendors shows an amazing result. In 2004 HTML5 development started [13] hence in 2005 performance was zero since the technology is still at its infancy. WAP was up and running as indicated in figure 7. It is important to note that by 1997 HTML4 had been released making HTML3 which uses pure HTTP request-response to move down-wards. From the chart WAP was on the rise from 2005 to 2007 and by 2008 HTML5 deployment was already at par with HTML3 but still far below WAP deployment [14]

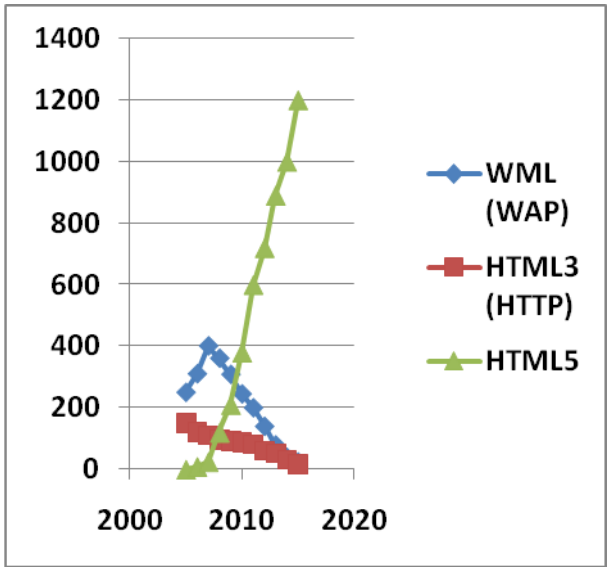

Figure.7. Plot of performance of the Technologies over the a period of ten years.

Due to the flexibility of HTML5 it continued a steady rise on mobile application development while WAP continued to decline as the years goes by and improvement continues in HTML5 mobile capability. HTML3 and WAP seem to continuously remained on the decline and many modern smart phones and PDAs do not have WAP specification for them at all. One may ask if that is the end of the road for the WAP technology or if Japan and Asia nations that still support WAP will do a technological reversal either in WAP, Dynamic WAP or any future WAP technology upgrade. Only the future can tell where the WAP technology is headed but the continuous improvement in HTML5 is making matter worse

since it seems to by integrating many other scripting technologies which were separate from the HTML developers have known.

5.1 WAP Performance Challenges

The challenge is that WAP (WML) still rely heavily on WMLS for dynamism and WMLS is only a subset of ECMAScript (JavaScript) which HTML5 developers are already trying to integrate into the main stream of HTML5 technology [15]. The one application for all mobile devices which HTML5 have achieved and is further improving upon make matter worse when compared to the variations in specification for different mobile devices which is used in WAP. It is even worse that many new devices do not have WAP specifications making difficult if not impossible for developers to adapt in those devices.

Moreover the time spent in developing the same application for different devices can be used in improving the single application that will run in different devices. The time may also be used in developing different applications which will make more fortune for the developers.

6. CONCLUSION

Mobile application developed with HTML5 technology will give better performance over simple HTTP web based mobile application and WAP based Mobile application. The support of HTTP5 in enabling Mobile application to run on cross platform efficiently means it will not have any constraint of any proprietary system. It also imply that resource constrained application which have required more resources like memory, processing will be run on mobile devices by using appropriate cloud services. The WAP technology seem to be at the verge of been abandoned by developers making the technology to be on its way to extinction if its supporters do not act very fast. The investment in its development and expansion will coolly go down the drain just as many investments in different obsolete IT technologies. Simple HTTP technology supported in HTML1 to HTML3 and even HTML4 may still remain for beginner developers and classic applications that lay on the internet for mobile devices which usage has continued to decline. W3C and mobile browser developers may not scrap the support of this versions immediately but there is a clear possibility in the nearest future that many mobile browsers may not continue to support them.

HTTP as an internet technology will continue to support HTML5 and even the newer technologies but they may no longer be the base for the architectural development of the mobile development technologies. In conclusion the research have analyzed selected mobile technologies and have presented their architecture with the aim of analyzing their strength and weaknesses and recommend actions to both technology vendors and mobile developers.

6.1Recommendation

In the light of the analysis and the result presented in figure 7 developers may have no option than to jump into the HTML5 band-wagon in their effort in developing web mobile application that will be efficient and developed once for majority of mobile devices. The growth of Android Operating System and its support for HTML5 support browser seem to add benefit of usage across different mobile devices powered by Android. Apple and Microsoft supported mobile devices are also not exempted. Developers investment in WAP development even if it is little my end up as an academic exercise if the vendors are not offering free and efficient tools and support. The lower versions of HTML have no future in mobile development but may still be very relevant in desktop development since many classic web site are not likely to go away in the nearest future and may still need to be supported by tomorrow browsers.

6.2 Future Work

In future, HTML5 will be very useful in the application field of mobile gaming, mobile healthcare, mobile commerce, mobile learning. Complex mobile application requiring high processor capability will be executable on mobile devices by leveraging cloud service for mobile application. We may need to find out whether the end of the road has come for WAP. Research can also be carried out on how to create a cloud computing application which uses Software as a Service for executing HTML5 application using Mobile Web Development Framework such as Phone Gap, Sencha, etc.

7. ACKNOWLEDGMENTS

Our thanks to the Oyol Computer Consult Inc and Fonglo Research Center for their contribution in typesetting and towards development of the work.

8. REFERENCES

[1] RFC2616 (1999) "Hypertext Transfer Protocol – HTTP/1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999. URL: http://www.rfc-editor.org/rfc/rfc2616.txt

[2] WAP-AS(2000) Wireless Application Protocol Architecture Specification,WAP Architecture WAP-210-WAPArch Proposed version 17-October-2000, Wireless Application Protocol Forum Ltd. (http://www.wapforum.org/what/copyright.htm).

[3] APION (2001) WAP Wireless Application Protocol, Apion Telecommunication Software Systems, The International Engineering Consortium http://www.iec.org

[4] IDC (2013). IDC Raises Tablet Forecast for 2012 and Beyond As iOS Picks Up Steam, Android Gains Traction, and Windows Finally Enters the Market. http://www.idc.com/getdoc.jsp?containerId=prUS23833612#.UV77q78pG3h. Accessed on 28.03.2015.

[5] Patrick M. (2012) Native versus HTML5 – where's mobile programming heading to? FHWS SCIENCE JOURNAL, Vol. 1, No. 1, 2013, 23-33, Germany

[6] Willnecker, F., Ismailovic, D. Und Maison, W. (2012). Architekturen Mobiler Multiplattform-Apps. Verclas, S. and Linnhoff-Popie, C. (ed.): Smart Mobile Apps - Mit Business Apps ins Zeitalter mobiler Geschäftsprozesse. Heidelberg, London, New York et. al. 2012. 403-418.

[7] Franke, F. and Ippen, J. (2012). Apps mit HTML5 and CSS3 für iPad, iPhone und Android. Bonn. 2012.

[8] Rama R. G., Mihai M., Bojan J. and Françoise P. (2012) HTML5 as an application virtualization tool 2012 IEEE 16th International Symposium

[9] Yang J., Zhang J. (2010) Towards HTML 5 and Interactive 3D Graphics 978-1-4244-8035-7/10 © 2010 IEEE

[10] Pavel S. (2012) Mobile development tools and cross-platform solutions 978-1-4577-1868-7/12©2012 IEEE

[11] Han Qi, Abdullah Gani (2014) "Research on Mobile Cloud Computing: Review, Trend and Perspectives".

[12] Nimit S Modi ,Proff.Yask Patel(2013) Web Interface using HTML5 for Interaction between Mobile Device & Cloud-Services, International Journal of Engineering Trends and Technology (IJETT) - Volume4Issue5- May 2013

[13] Kevin C., Aaron B., Gavin F. (2012) HTML5 and the Mobile Web, International Journal of Innovation in the Digital Economy, 3(2), 40-56, April-June 2012

[14] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang (2015) A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches http://onlinelibrary.wiley.com/doi/10.1002/wcm.1203/abstract

[15] Corral, L.. (2011). Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm. Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software. New York. 2011. 181-183.

9. ABOUT THE AUTHORS

Eke Bartholomew PhD, MCPN, MACM, FIPMD is a Software Engineering / Computer Science Lecturer at the University of Port Harcourt and Mobile Application Developer in Oyol Computer Consult Inc. His research interest is in SE Methodologies and Mobile IT deployment.

Dr. Onuodu, Friday E. is a Lecturer at the University of Port Harcourt. His research interest is in Data Mining and Data Extraction using both Mobile Devices and Desktops. He also has interest in IT deployment analysis. He has many publication in learned journals.