Unhosted Email Client

[#1]Rukshan Chathuranga, [#2]Sashrika Waidyarathna, [#3]Kanthasamy Sureshkumar, [#4]Kanagasabai Erlilverl, [#5]Prof. Gihan Dias
[#]Department of Computer Science and Engineering
University of Moratuwa, Sri Lanka
[#]1-5{rukshan.10, sashrika.10, suresh.10, erlilverl.10, gihan} @cse.mrt.ac.lk

***Abstract*— We developed a new email client which runs on the browser and is not dependent on a particular server, based on the *Unhosted* web application architecture. The client can connect to any standard mail server directly from the browser, using IMAP and SMTP protocols so that the user is not limited to the webmail clients provided by various email service providers. As our client runs in a browser, it is independent of the operating system. It provides a uniform user experience on any mail system. As a part of our unhosted email client, we have developed a browser extension to allow HTML5 clients to connect to IMAP and SMTP servers. Using this extension, developers may introduce new e-mail clients which are both server and operating system independent. Our client application also provides offline features so that the user can load the application and read/create mail even if he/she is not currently connected to the Internet.**

***Index Terms* – email, HTML5, IMAP, offline, Unhosted, webmail**

I. INTRODUCTION

Accessing email via a web browser is very common today, and the use of desktop e-mail applications has declined. All webmail clients that we use today are hosted within particular mail servers. For example, Gmail is hosted in Google mail servers and Outlook.com is hosted in Microsoft mail servers. We cannot login and read emails from one mail server using another webmail client (however, some e-mail services can retrieve mail from other servers).

But today users tend to have more than one email account with several mail service providers. Therefore when using webmail clients users may need to switch between email clients for each mail service. So they have to cope with certain pros and cons of each web mail clients.

We implemented Unhosted email client following Unhosted web architecture [1]. Unhosted web application architecture is a new emerging web app architecture. Also known as server-less, client side web application architecture. Developers develop the front end web applications using HTML5, CSS, JavaScript so that they do not need a server to run its application logic. These web applications are not tied up with any particular backend and try to break the web monopoly [2]. End user can choose which back end he wants to connect in a given instance.

Our email client is the first full implementation of an Unhosted email client web application. We used standard SMTP, IMAP protocols to communicate with the mail servers directly from the browser. Raw socket connections cannot be opened using JavaScript through the browser due to security concerns [3] [4]. If raw sockets were possible, then it would be compromising the sandbox that web browsers have carefully constructed to protect its users. Then the web page could be surreptitiously sending users private and sensitive content from user's device back to some third party server.

As a solution we used browser plug-ins to open socket connections to mail servers. Browser plug-ins have enough privileges to open socket connections to any servers. The browser plug-ins we implemented are lightweight and they are only used to open connections to mail servers. All processing which is specific to the application is done outside the plug-in scope using JavaScript.

Our goal was not to come up with an email client with new features or facilities. So as a startup we used Horde IMP web mail system as our front end email client. We extracted views and user interfaces from Horde IMP project. Most of the JavaScript codes of the Horde was reused and additional controllers were written to process emails and communicate with servers.

Since this runs on the browser, it is platform independent. This email client can be used on any operating system with a supported browser. Offline features are implemented using HTML5 Application Cache API and HTML5 IndexedDB API. There are two way that one can use the unhosted email client. One way is loading the source files through the network. You can host the JavaScript source files and other resources in a server. Once you load the application into the browser through the network, it will download and cache all the resources required to run the application. All the mails will be stored in IndexedDB inside the browser. So once emails get stored in the IndexedDB, user can access the emails without having an internet connection. Also application can be copy and transferred to another machine for installations. Application can be used as portable emails client where opening index.html file from a supported browser, application will launch and run as usual.

978-1-4799-1740-2/15/$31.00 ©2015 IEEE

II. APPROACH

A. *Current Systems*

Even though there are enormous numbers of webmail clients available, there is none which is highly decoupled with the backend mail servers. There are applications which can connect to different mail servers through single entry point. But they have intermediate components hosted in a server. One such implementation is SocketHub[5]. The email client has to have an intermediate server running remotely to relay the requests. SocketHub has implemented a intermediate server to used relay messages. Sockethub server handles TCP communication to IMAP/SMTP servers. In the same time it enables Web socket communication from browser to intermediate server.

A proof of concept for an IMAP client library written in JavaScript which runs as a Chrome app, is another system closely reflecting the idea [6]. This has written on top of node.js using node's IMAP client library. To open a raw TCP connection to a server it has used Chrome's socket API. Since this is a proof of concept, the app itself only supports mail fetching part. The application also supports sending mail part though it is not implemented in the app user interface, but the internal functionality exists. This application has lot more works to be done to make it usable. Since this is implemented as a Chrome app, this application doesn't run in any other browsers.

A REST based email system has also been implemented. There, the implementation includes the entire email system. When paying attention to the client component, it communicates with its mail servers using REST based services [7]. That is possible as the usual IMAP and SMTP servers are wrapped by a module which intermediate the communication by mapping REST to IMAP/SMTP protocol communication. This email client supports offline features as well. Its mail servers provide REST APIs for developers to come up with their own client implementation. But, still, it will only work with that system as most of the other email providers do not give REST based services.

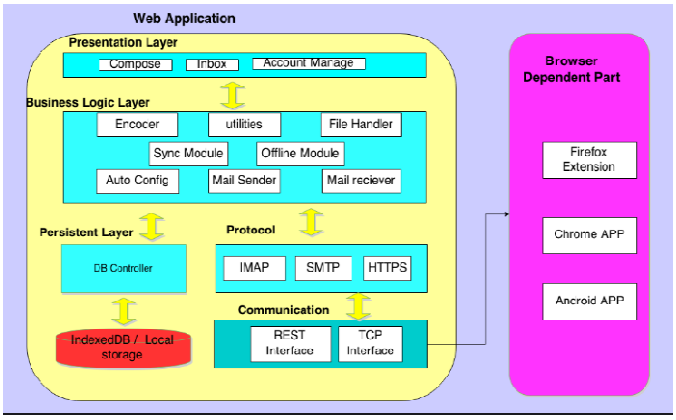

Fig. 1. Overview Architecture of Unhosted email client

B. *Unhosted Email Client Application*

Unhosted email application is an email client entirely running on browser with the support for offline mode functionalities. The application has been entirely built using HTML5, JavaScript and CSS. As the name implies, the application doesn't have to run on a server. The running context of the application is fully restricted to browsers' environment. Dependency is there with browsers but not with OS platforms. This application supports modern browsers such as Chrome and Firefox. Firefox needs a custom extension to help open the raw TCP connections. As in the latter, Chrome needs chrome app. Extension and Chrome app are using APIs provided by the respective browsers. The application uses IMAP and SMTP services provided by mail servers using respective standard commands.

User can connect to any mail servers as the backend, to fetch and send mails using the application. The application utilizes the application caching features introduced by HTML5. And IndexedDB is used to store application data within the context of browsers for the offline usage. The application frequently requests the mail servers, when connected, to fetch the updates of messages. The communications between application and the mail servers are managed by extension/chrome app.

1. *Synchronization*

Synchronization is one of the major functionality of an email client and it should be available in any email client. But unlike a hosted webmail client, a hosted webmail client and desktop email client must provide optimized synchronization mechanism [11].

Thunderbird mail client provides completely different synchronization implementation for IMAP [8]. Similar to the Thunderbird synchronization, Whole Folder sync, Partial folder sync mechanism used in this Unhosted webmail client as well. Also to optimize synchronization further for RFC 4551 supporting Server, HIGHESTMODSEQ based synchronization is used [9].

Unlike the desktop email client, webmail clients have problem with limitation of the storage. To address this issue, unhosted email client does not sync all the mailboxes or all the mails inside one mailbox. It saves predefined number of emails and that value can be changed according to the space limitation. When sync happening old mails are marked as "removed" in local database and remove body and attachment to accommodate new emails. When user selects such emails from a mail box those are fetched again temporarily and shown.

As future improvements sync time push email mechanism is expected to be implemented. But it is still in the research

level and current version doesn't support push email. Also to optimize storage availability, use of browser file system sandbox for attachments store, expected to be used instead of IndexedDB database.

2. *Offline Capabilities*

A common drawback in all modern webmail clients is the lack of support for offline use. If you do not have a working internet connection you cannot load your emails and cannot load your webmail client. So that the users may not be able to utilize his/her time to read and respond to emails.

To address this problem, we used the capability of HTML5. HTML5 introduced Application cache API which enables to build offline version of a web application, by creating a cache manifest file. Once the user loads the web application into the browser, all HTML, CSS, JavaScript files are cached within the browser, so that the user can load the application even without the internet connection. Also we used Indexed Database API which was also introduced by HTML5, to store emails within the browser.

With this architecture, new email client provides offline browsing which users can use the client when they are offline. Also the email client will load faster than the other webmail clients because cached resources loads faster. This will also reduce the server load because browser will only download updated/changed resources from the server [10]. Emails and attachments will be stored within the browser and only the new emails will be downloaded from the server. All the mail operations can be performed in the offline mode and those operations will be done on the server once the browser connects to the internet.

There are few web storages we consider as the database. i) Web SQL database ii) Local storage iii) IndexedDB. IndexedDB is useful for storing larger amounts of structured data which is a JavaScript based object oriented database. But Firefox has no limit and upper bound of the storage is the free space in the disk. In Chrome, it creates the IndexedDb inside the shared pool. The shared pool can be up to half of the of available disk space. Each app can have up to 20% of the shared pool. Once the storage quota for the entire pool is exceeded, the entire data stored for the least recently used host gets deleted. So there is a potential risk that the user local email can get deleted without a prior notice if available storage space shrinks. But no damage will be done to the emails on the server.

3. *Presentation*

Unhosted email client has very simple presentation layer which was built on top of Horde IMP open source webmail user interfaces. This contains set of HTML, CSS and JavaScript files. Most of them are extracted from Horde [12]. In presentation all the views are implemented to support good user interaction with mail client. Which include separate browser window to compose, reply and forward mail. Also Inbox view with multiple mailboxes. Also application has provided the interface to change application default behavior and make changes to configuration. Mail user accounts handling interfaces are provided to the users. Users can configure email client with their mail accounts and settings for each mail accounts.

4. *Mail Servers Communication*

Browsers are unable to directly communicate with IMAP/SMTP servers due to security reasons. Also no inbuilt JavaScript APIs are provided by browsers to open TCP connections to servers except Firefox OS. The reason is Firefox is not running inside a sandbox environment. The solution would be invoking middle servers similar to the Node.js. But such a server needs an installation on client device before it can be used. Thus without going on any complex installation and platform independent installation we used browser extensions to communicate with servers.

These extensions are light weight middle component which can open a socket connection to any servers via TCP/IP protocol. Also it supports secure, non-secure communication and SMTP security (STARTTLS) as well. At the first run of Unhosted webmail client it asks user permission to install the extension/app on client browsers. It is lightweight and takes small amount of time to install on the browser.

As extension for Firefox, Chrome has two separate execution environments. According the chrome, extension are not allowed to open TCP connection to random endpoints. Only Chrome apps can do it. Chrome apps are provided with some extra APIs [13] to utilize the socket connection from the chrome browser. Meanwhile Chrome apps can also communicate with web application running on the browser. Web application has to initiate the communications. Web application uses the Chrome app Id to send messages to the relevant chrome app. Chrome app stimulates to the events from the given URLs. URL/URL pattern is provided in the manifest file to identify and accept events from it. Firstly, the chrome app opens a connection to mail servers on request and make them secured. Each time it gets a message; it extracts the IMAP/SMTP command and send it to the mail servers through opened secured connection. Then whenever it gets the response, it buffers the response until the end of the response arrives. Then relays it to the mail application back.

For Firefox it followed bootstrapped [14] extension which is run automatically when Firefox browser startup. It will be started without user interaction when user opens the application. Extension is implemented such that it is compatible with all the modern browsers including Mobile,

Desktop and Tablet devices. Since Firefox has provided powerful API for extension development, Those APIs are used to implement socket connection to the servers. Inside the extension it keep the list of socket connection instance to provide concurrent socket connection to the servers. All the communication happens between the application and extension used the Socket connection ID. Extension identifies this Socket connection ID and if there is connection for that ID used it. If not new socket connection is created and assign to it. With this it is possible to provide multiple server communication without jamming data each other.

In generally all of this Extension/App follows same architecture and functionality. Only the implementation is different. Those are used browser support API and related communication pattern. All of these modules provide same interface to the application. That is functions to send, receive request and response from the servers. Also these modules are responsible to provide only the needed information to the application. That means when a request says it need to fetch all the mail Id of inbox mailbox, extension responsible to give the response with mail IDs. If something wrong with connection and invalid request, extension shouldn't provide the unnecessary information except error message. It is responsible to provide complete response from the servers. Because when a response has large information it gives the data by parts. In that case it is responsible to buffer complete response and send back to the application. To do this job, application provides the necessary information to identify the response. It is done by regular expressions and those expressions are provided to the extension when the request is send to the extension.

For Firefox and chrome it follows the same architecture. But for the other browser this architecture can be different. For android devices, browser those are not support extension implementation now. To support that we expected to implement android application to start the browser. But even this application also follow the same architecture, functions and responsibilities mention above. Thus application has predefined Interface for these modules (Extensions/Chrome App and Android app). With this, it is possible to extend the application usability on most of the browsers like android browsers and IOS browsers.

5. *Auto Configuration Module*

Most of the email clients face a problem of setting server configuration information in client application. Since our client can add different email accounts, each account may have different server configurations. As a solution this application has implemented a way to automatically detect the configuration settings based on the email address. Thunderbird mail client uses several ways to automatically configure the mail client [15]. In this, it used Mozilla ISPDB to set the configuration and for that it uses HTTPS request to the Mozilla ISPDB. For that, it is impossible to use HTTPS request from HTML pages directly due to CORS policies. Thus it also uses TCP interface to request those information. Also, we expect to implement other mechanisms to improve the auto configuration.

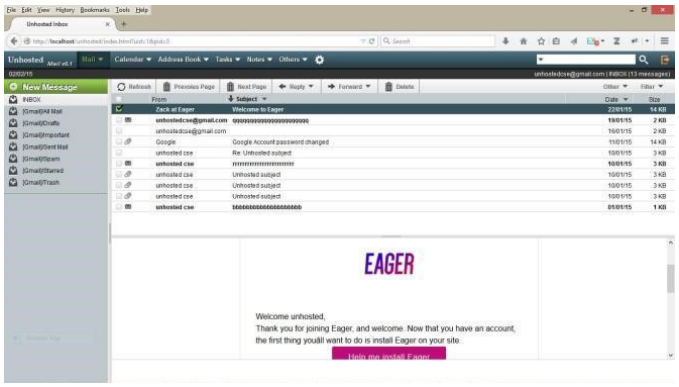

Fig. 2. Unhosted mail client on Firefox running on windows

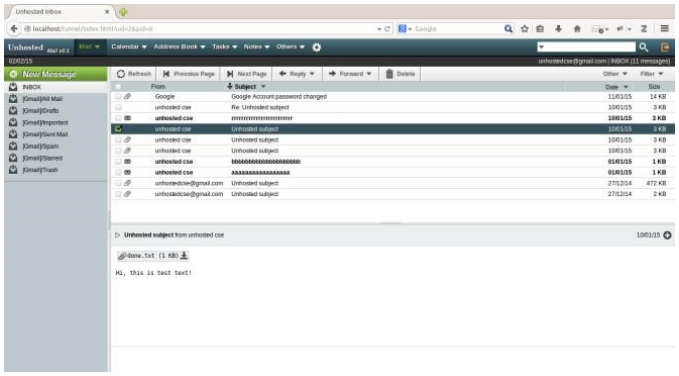

Fig. 3. Unhosted mail client on Firefox running on Ubuntu

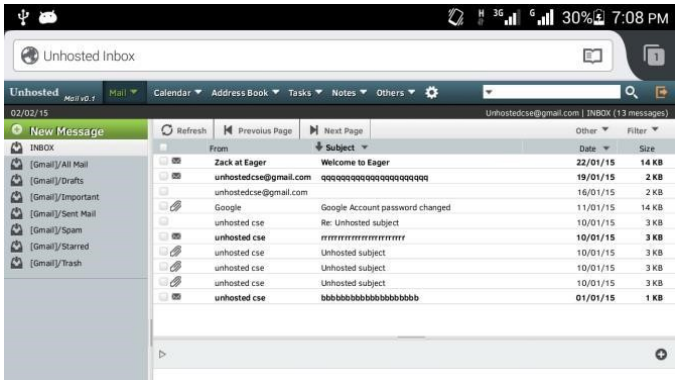

Fig. 4. Unhosted mail client on Firefox running on android

III. RESULTS

The Unhosted webmail client works on all current Firefox browsers running on any platform. But it runs only on desktop version of the Chrome browser. Users can do all basic email

functions like sending, receiving, syncing, attaching files, replying, forwarding and address book usage. Users can add any number of email services to this email client and it is possible to use them in parallel on different browser instance or tabs. Currently it works with IMAP/SMTP mail servers including Google, Microsoft and Yahoo as well as the UoM mails. It has been tested on other Postfix servers as well.

Application has tested on several devices on different platforms. The screen shot of the application run on these devices are shown on Fig. 2, 3 and 4.

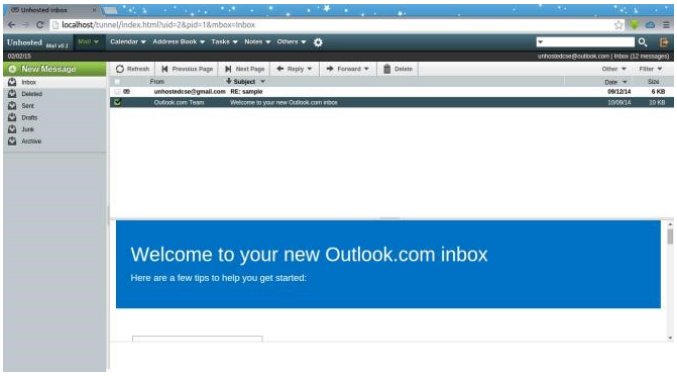

Fig. 5. Unhosted mail client on Chrome running on Ubuntu

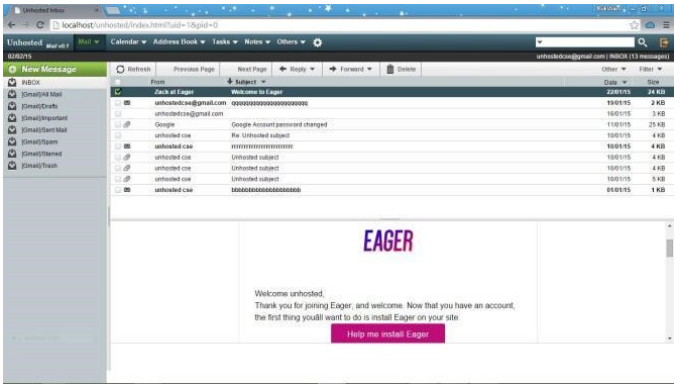

Fig. 6. Unhosted mail client on Chrome running on Windows

It also tested with sample users with their email accounts and positive feedback has been received from them. The time to take synchronize one mailbox, having emails with attachment, is considerably fair. But sync optimization based on push email is still on the research level and hope to include it in the next version.

Currently application runs only on desktop versions of chrome browsers due to limitation of extension development. The screen shot of those instances are shown in Fig. 5 and 6.

IV. GENERALIZABILITY, LIMITATIONS AND FUTURE WORK

One limitation of this Unhosted email client is the dependency of browser plug-in. Currently only Chrome and Firefox browsers support plug-ins which are capable of opening TCP socket connections to servers. But if other browsers start supporting this feature, we only have to write a thin plug-in which opens connections to servers. All other application logic is browser independent and written using JavaScript.

Currently this application doesn't support additional security over the browser database. User emails are stored in the IndexedDB within the browser in plain text. Thus we do not recommend using this application on shared computers. Currently we assume, if a person has logged into his machine using his/her operating system account password, he/she should have access to this application without any further authentication process. Though security over data can be implemented by encrypting the database using account password as future enhancements.

Mail synchronization with the local database and server can be done in more efficient way. Currently we haven't considered performance of mail synchronization much. Current version of Unhosted mail client ships with an address book module. As future enhancements more modules hope to integrate with the system in next versions.

To generalize this email client without depending on the browser plug-ins, there is a working draft in W3C for TCP/UDP raw socket APIs. This draft was proposed very recently and once this was implemented in all browsers as W3C standard, we can use this Unhosted email client over all browsers with a very minor modification.

V. CONCLUSION

In this paper we introduced Unhosted Email Client which is a web application which runs on the browser, without depending on a particular backend server. Frontend email client is independent of backend mail servers. Therefore it can be configured with any SMTP/IMAP server as the backend. Since modern browsers do not allow to open TCP socket connections to servers, we have developed browser plug-ins/extensions to securely open TCP socket connections. Currently this supports both Chrome and Firefox browsers. As the front end email client we have extracted user interfaces from Horde IMP open source project. Using HTML5 features, we have made this email client offline enabled. Therefore users can read and respond to emails even if they do not have a working internet connection.

REFERENCES

[1] Unhosted.org, “unhosted web apps”. [Online]. Available: https://unhosted.org. [Accessed: 27- Mar- 2015].

[2] J. Presbrey, “Linked Data Platform for Web Applications”, M.S. thesis, Massachusetts Institute of Technology, 2014.

[3] C. Nilsson, (2014, Dec 02). TCP and UDP Socket API [Online]. Available: http://www.w3.org/TR/raw-sockets/.

[4] kscarfone, (2014, Jan 19) “TCPSocket,”. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/TCPSocket.

[5] Sockethub, “Sockethub - a polyglot messaging service.”, 2014. [Online]. Available: http://sockethub.org. [Accessed: 27- Mar- 2015].

[6] “Javascript IMAP email client for browsers”, 2013. [Online]. Available: http://www.hiddentao.com/archives/2013/08/15/javascript-imap-email-client-for-browsers/. [Accessed: 27- Mar- 2015].

[7] Dias, G. Karunarathna, M. Udantha, M. Gunathilake, I. Pathirathna, S. Rathnayake, T. REST based offline email system. in Proceedings of the Asia-Pacific Advanced Network, (2012), 85-92.

[8] “Gaia/Email/Implementation/MailSynchronization – MozillaWiki”, 2013. [Online]. Available: https://wiki.mozilla.org/Gaia/Email/Implementation/MailSynchronization. [Accessed: 27- Mar- 2015].

[9] Wiki.mozilla.org, “Thunderbird:IMAP RFC 4551 Implementation – MozillaWiki”, 2009. [Online]. Available: https://wiki.mozilla.org/Thunderbird:IMAP_RFC_4551_Implementation. [Accessed: 27- Mar- 2015].

[10] R. Karthik, D. R. Patlolla, A. Sorokine, D. A. White and A. T. Myers, 'Building a secure and feature-rich mobile mapping service app using HTML5: challenges and best practices', in MobiWac '14 Proceedings of the 12th ACM international symposium on Mobility management and wireless access, 2014, pp. 115-118.

[11] Xu, Fengyuan, Liu, Yunxin, Moscibroda, Thomas, Chandra, Ranveer, Jin, Long, Zhang, Yongguang, and Li, Qun. Optimizing Background Email Sync on Smartphones. in Proceeding of the 11th annual international conference on Mobile systems, applications, and services.

[12] Horde.org, [Online]. Available: http://www.horde.org/licenses/gpl. [Accessed: 27- Mar- 2015].

[13] Chrome, “Chrome.Socket” [Online]. Available: https://developer.chrome.com/apps/socket.

[14] Makyen, “Bootstrapped extensions”, Mozilla Developer Network, 2014. [Online]. Available: https://developer.mozilla.org/en/Add-ons/Bootstrapped_extensions. [Accessed: 27- Mar- 2015].

[15] B. Bucksch, “Autoconfiguration in Thunderbird”, Mozilla Developer Network, 2015. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Thunderbird/Autoconfiguration. [Accessed: 27- Mar- 2015].

[16] “HTML5 Application Cache.” [Online]. Available: http://www.w3schools.com/htmL/html5_app_cache.asp.