Internals of Firefox Mobile Operating System with NFC Components

Saminath

* Accenture, Bangalore

Abstract- Firefox is the emerging mobile operating system from Mozilla Gecko Technology. It creates the new revolution for the handheld device product development. Open source operating system has great support to platform integrator and power full to the mobile Application developer. This paper highlights the Layers of the Firefox operating system with details of layered architecture system. In general this explains about the How Android BSP is coupled with Firefox Layers. It briefs about the Firefox specific changes on the Android BSP. Detailed explanation on the NFC components, NFC data structures, NDEF messages structure and NFC daemon in Firefox system. This paper elucidate the requirement of Firefox operating system and constraint on Hardware and software. Currently, Firefox Application runs in various target systems like, simulator, Emulator, Desktop System, Mobile Phones, Hardware Development platform, and Dual boot support with Android OS.

Index Terms- NFC: Near Field Communication, HAL: Hardware Abstraction Layer, NDEF: NFC Data Exchange Format, I2C: Inter-Integrated Circuit communication protocol

I. INTRODUCTION

Developer and consumer have very much expectation in Firefox, because of its potential and power of the open source community. Currently Firefox flavor comes only in 'Smartphone'. The entire features are packed into single device other than else. The features are Social media supports in face-book, twitter, LinkedIn, Video Editor Software's are pre-installed as free applications in the device. Maps supported from HERE-Maps application get the traffic information, local transit information and direction of the movement, apart from that music gallery support, Browser supports, SMS authenticating for mobile billing, conference calling. Those are HTML-5 based application which is ready to work on the phone, not much customization required for porting into Firefox OS. Primary problem is lacking of shifting Android based Mobile OS to New Mobile Os with Linux BSP, "What is Firefox Layer Interaction with Android BSP", How the Application is exactly running on the Firefox, Interaction among the Firefox software layers and how to bring the Firefox operating system on alien devices.

The high-level view of the Firefox system is broadly divided into two segments, User Space and Kernel Space, based on the permission, process implementations. The kernel space has restricted access, physical communication with hardware system, driver for peripherals and exposes the functionalities to user space. User space program primary task is to get the access of kernel services, facilities for GUI framework; end user application which consists of core libraries provides to the API interface and other miscellaneous services.

II. FIREFOX KERNEL-SPACE

This layer is completely hardware related code and configuration of the peripherals. The primary source codes are u-boot code, Linux kernel, firmware, and drivers. Firefox reuses the kernel from Android; No additional features required identical with Android base source code. Feature of the kernel is IPC mechanism for RPC, optimized logging mechanism, mapping large memory to user-space; kill least recently used process and sophisticated power management system. Kernel offers the core services such as graphics, security, networking and memory management.

A. *Boot strap:* ROM Code Checks if a valid application is present in booting sector and downloads it into internal SRAM. Bootstrap change hardware configuration, downloads U-Boot binary and starts the Boot Loader.
B. *U-boot:* The boot loader, responsible for downloading kernel images, configures the network, SD card, etc. Then it loads the Device Tree Binary and starts the Linux kernel. Enhancement of fast boot supports firmware up gradation in USB interface.
C. *Linux Kernel:* Primary task is process management, Memory management, Network stack management, and power management, device driver for processor specific and peripherals and scheduling the tasks, it requires the Linux kernel features like ASMEM, BINDER, LowMemoryKiller (LKM), Logger, USB Gadget, and ION. These android additional driver in Linux kernel released by GPLv2, offering extensive service to user space.

D. *HAL (Hardware Abstraction Layer):* This is middleware layer, between the kernel and user space. It abstracts device specific routines, algorithms and exposes the well define interface to application layer. It has various task routines
    1) Module loading and unloading.
    2) Separate the platform logic from hardware interface.
    3) Communication channel for Application framework to access the devices

The open source projects like libusb, BleuZ, GPS and camera user space libraries are integrated in HAL. These libraries are low level access to across multiple hardware platforms. HAL loads the component into two ways, In runtime using dlopen() and automatically loaded by the dynamic linker. Hardware Platforms are Processor with Hardware specific components like Codecs, Modems, Display Device, and Sensors inter-connected with wires.

III. FIREFOX USER-SPACE

Like Android, Firefox is built upon the open source platform, Bugzilla releases all the source code to community to make available. Entire user space application layer spilt into three domains, Application, Open web platform interface, Infra structure layer. It ensures that each layer has its own security; other applications can still interact with each other. It has support of Native Libraries, Multimedia Frameworks and Daemons.

A. *Application Layer*: In this layer component consist of Java Script, HTML, and CSS as a regular web application framework. Firefox uses the gecko (like webkit in android) powerful and standard-compliant browser engine. Gecko engine is attractive for browser application developers, its streamlined code base, supports W3C standards, and open-source.

B. *Gaia*: The user interface of the Firefox OS platform. Anything we need to draw on the screen, Firefox OS start up the Gaia layer. Gaia implements the lock screen, home screen, and all the standard applications which we expect on a modern smartphone. Gaia is implemented entirely using HTML, CSS, and JavaScript. Its only interfaces to the underlying operating system are through open Web APIs, which are implemented by the Gecko layer. Third party applications can be installed alongside the Gaia layer.

C. *Power of HTML5 and JS libraries*: Firefox supports HTML5 features; it makes much more capability of the web application development. Various built in application like browser, calendar, camera, contact manager, dialer are installed on the device. Firefox supports various libraries in the Gaia layer. Currently Firefox OS Gaia uses a modified version of Fabien Cazenave's L10n.js library to localize the default Apps that are available in Firefox OS. It is available in the Gaia source tree. The L10n.js parser also supports import rules that can be used for client side language selection.

D. *Security system Sandbox*: The Firefox OS security framework uses sandboxing as a defense-in-depth strategy to mitigate risks and protect the mobile phone, platform, and data. Sandboxing is a way of putting boundaries and restrictions around an app during run-time execution. Each app runs in its own worker space and it has access only to the Web APIs and the data it is permitted to access, as well as the resources associated with that worker space (IndexedDB databases, cookies, offline storage, and so on). For details, check the Firefox links

E. *WebAPI:* It is a term used to refer to a suite of device compatibility and access APIs that allow Web apps and content to access device hardware (such as battery status or the device vibration hardware), as well as access to data stored on the device (such as the calendar or contacts list). By adding these APIs, it helps to expand what the Web can do today and only proprietary platforms were able to do in the past.

F. *Gecko Runtime:* This is the Firefox OS application runtime; this layer provides all the support for the trifecta of open standards: HTML, CSS, and JavaScript. It makes sure those APIs work well on every operating system Gecko supports. This means that Gecko includes, among other things, a networking stack, graphics stack, layout engine, a JavaScript virtual machine, and porting layers. Gecko is combination of both browser engine and rendering engine.

G. *HTML API:* There are many HTML API available to develop the code for web using Java script.
    1) Connectivity: allowing you to communicate with the server in new and innovative ways.
    2) Offline & Storage: allowing webpages to store data on the client-side locally and operate offline more efficiently.
    3) Multimedia: making video and audio first-class citizens in the Open Web.
    4) 2D/3D Graphics & Effects: allowing a much more diverse range of presentation options.
    5) Device Access: allowing for the usage of various input and output devices.
    6) Styling: letting authors write more sophisticated themes.

H. *Gonk* : Gonk is the lower level operating system of the Firefox OS platform, consisting of a Linux kernel and users pace hardware abstraction layer (HAL). The kernel and several of the user space libraries are common open-source projects:

Linux, libusb, bluez, and so forth. Some of the other parts of the HAL are shared with the Android project: GPS, camera, and others. The Gonk is a very simple Linux distribution from open source. Gonk is a porting target of Gecko; The port of Gecko to Gonk, is just like a port of Gecko to Mac OS X, Windows, and Android. Since the Firefox OS project has full control over Gonk, we can expose interfaces to Gecko that can't be exposed on other operating systems. For example, Gecko has direct access to the full telephony stack and display frame buffer on Gonk, but doesn't have this access on any other operating system.

I. *Init process*: It is entry point of Firefox OS. Firefox init process executes the init.rc and init.b2g.rc configuration and starts up the b2g process. During start up this process handles the file system mounting, runs various system services, finally runs the b2g in highly privileged task.

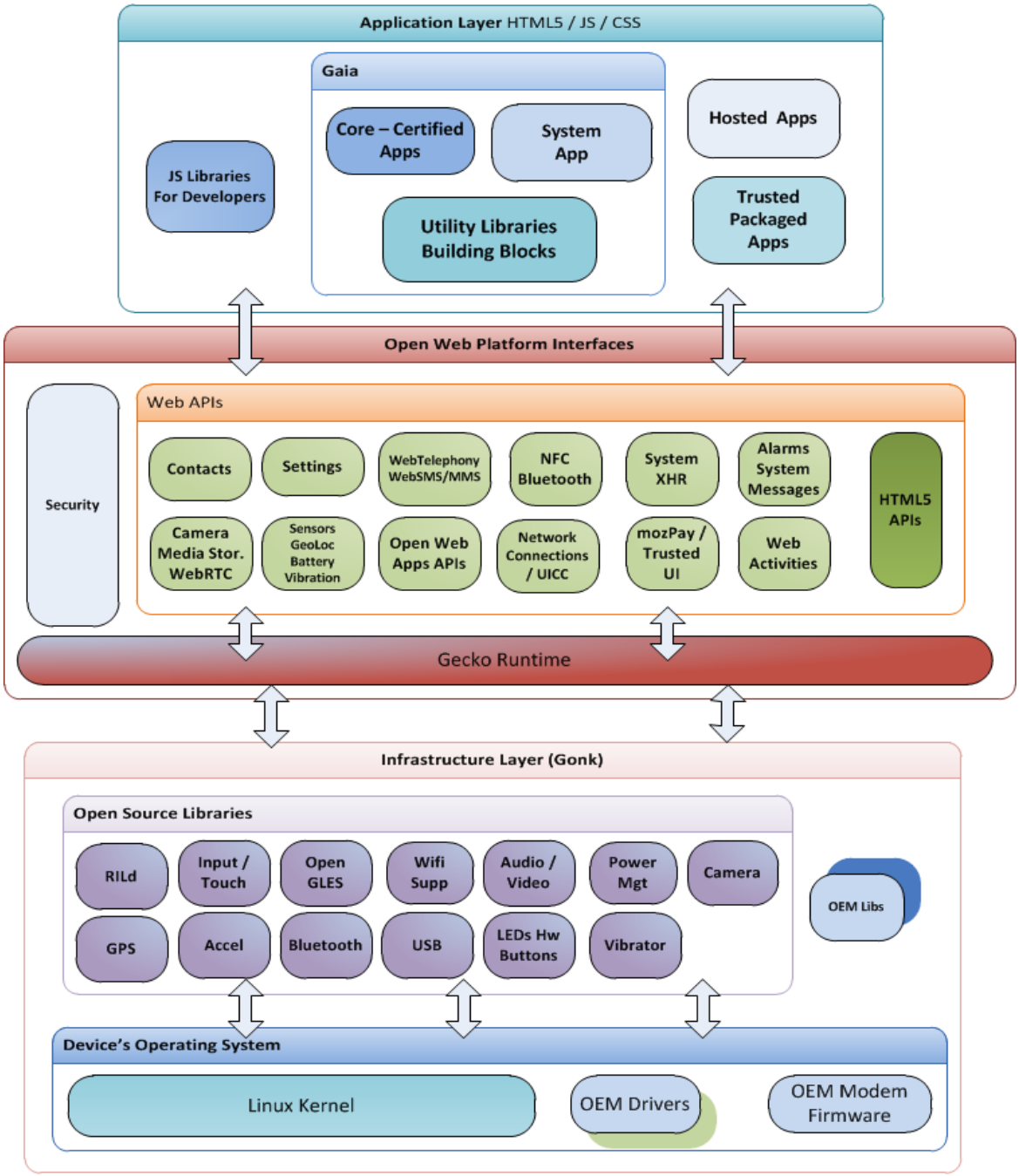

Figure -1: Architecture diagram of Firefox platform by Mozilla

IV. UNDERSTANDING OF NFC COMPONENT IN FIREFOX

User can share information to another NFC enabled device (like Mobile phone and Tablet) communication via paired device. Share content may be contact information, Image content, Video content, music content, URL, NFC enabled payments. The data can be shared via NFC to another device by tapping the mobile device with another NFC enabled device. Transfer of content takes place directly over NFC or support by other wireless communication (like BT and WiFi)

There are four types of NFC forum tags presents, which is build top of the product. Below table provides the NFC Forum tags and corresponding available product.

Table-1: NFC Tags Types

NFC Forum Tags	Compatible Products
NFC Forum Type 1 Tag	Innovision Topaz
NFC Forum Type 2 Tag	NXP MIFARE Ultralight
NFC Forum Type 3 Tag	Sony Felica
NFC Forum Type 4 Tag	NXP SmartMx and DESire

NDEF (NFC Date Exchange Format) is the encapsulated messages exchange between two NFC Forum devices. NFC Forum TAG is combination of encapsulated NDEF messaged and NFC forum Type TAG. This information stored as application data inside the NFC device.

Table-2: NFC States

State	Description
No Tech	No NFC-compatible technology discovered
Tech Discovered	A NFC- compatible technology has been discovered. This includes a NFC-A tag, a tag (possible different technology) with a NDEF message, or another NFC-enabled device
NDEF Details Pending	A tag or P2P device with a NDEF message has been discovered and the application requests the meta-data of that message. The details request is pending and not yet completed
NDEF Discovered	An NDEF-compatible tag has been discovered
NDEF Connect Pending	Applica5on wants to connect to the NDEF compa5ble tag. The connect request is pending and not yet completed
NDEF Connected	Applica5on is connected to NFC-A compa5ble tag
Close Pending	Applica5on is closing the connec5on to the tag. The close request is pending and not yet completed
NDEF Write Pending	Applica5on has issued a NDEF message write request that has not yet completed
NDEF Read Pending	A tag or P2P device with a NDEF message has been discovered and the applica5on reads that message. The read request is pending and not yet completed
NDEF Make Read Only Pending	Make the tag holding a NDEF message read-only. The request is s5ll pending
NfcA Tag Details Pending	A NFC-A compa5ble tag was discovered and the applica5on requests the tag details. The tag details request is pending and not yet completed
NfcA Tag Discovered	A NFC-A compa5ble tag has been discovered
NfcA Tag Connect Pending	Applica5on wants to connect to the NFC-A compa5ble tag. The connect request is pending and not yet Completed
NfcA Tag Connected	Applica5on is connected to NFC-A compa5ble tag
NfcA Transceive Pending	Applica5on issued a transceiver request. The request is ending and not yet completed

NFCD is a daemon in user space layer, it provides access to the NFC chipset. It uses IPC to communicate with Gonk layer. The scope of the protocol tag is reading and writing of NDEF messages, NFC-A (ISO 14443-3A) compatible tags (such as NXP's Mifare Ultralight tags) and P2P.The protocol is as similar as described by Android. NFC has two stack required to run the application. NFC kernel stack has driver for the NFC chipset, mostly comes with I2C protocol data communication. HAL implementation is driver commands, passing with register call-back implementation. Low level hardware control library for each chip specific in the NFC subsystem. Major task in the user space NFC daemon are,

1) Reading and Writing TAG specific Handling
2) LLCP protocol
3) NDEF protocol parsing and Decoding
4) Receive the NFC Notification and handover to application
5) Register content of target and create mozNfcPeer Object

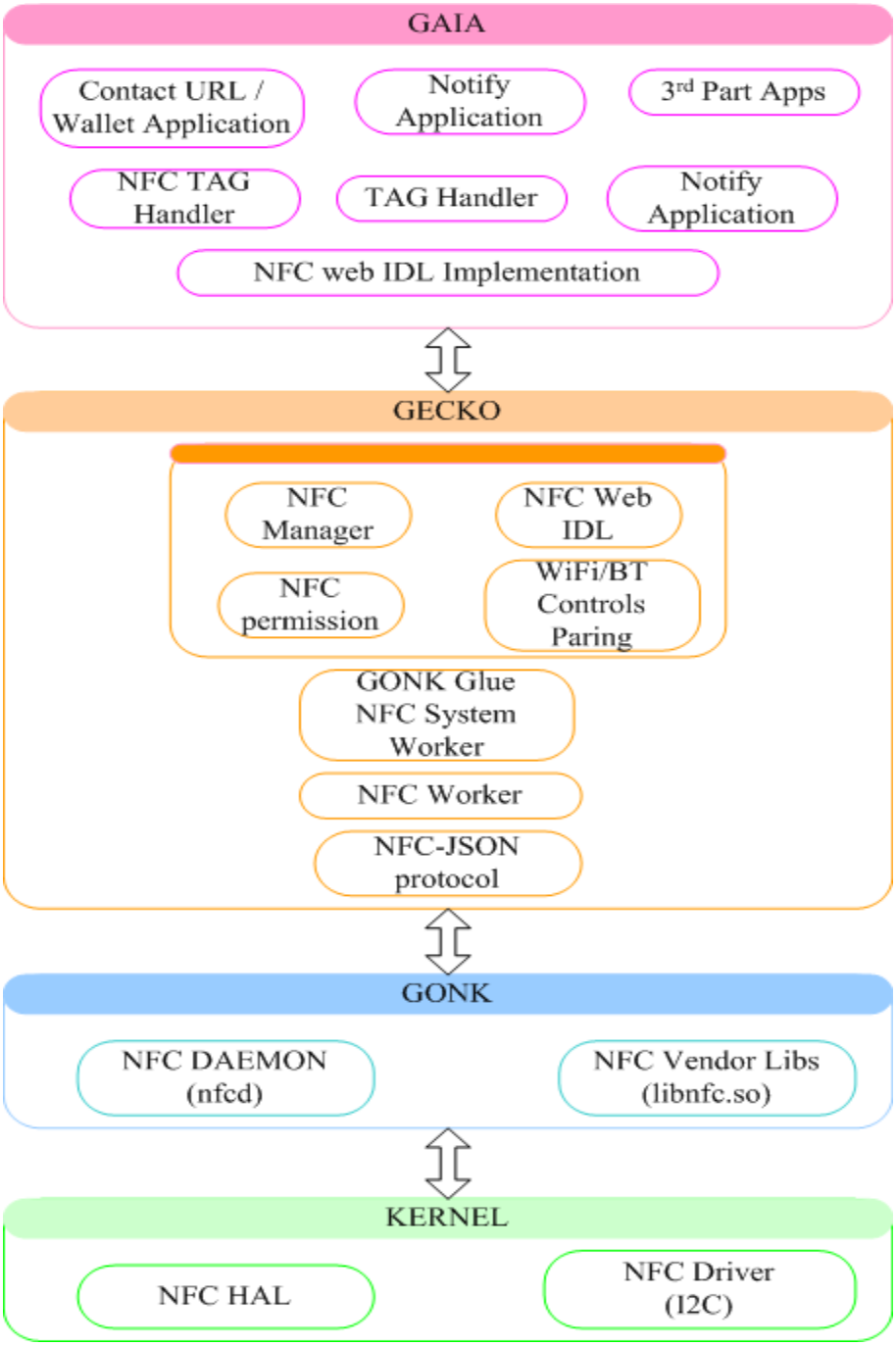

Figure -2: NFC Architecture diagram in Firefox platform

Table-3: NFC Software source Tree in Firefox

Source Tree Location	Implementation details
drivers/nfc/bcm2079x.c	NFC Low level driver implementation. It uses I2C based protocol to communicate with hardware chip.
gecko/ipc/nfc/	It provides communication between the nfcd (daemon) and Application Layer
dom/system/gonk/	Implementation of NFC worker component. Provide high-level API representing NFC capabilities. Responsible for converting NFC requests from Content process to binary data NFC Responses from binary data to dictionary objects.
dom/nfc/	Implementation of WebIDL, Gaia Application uses the function of read/write NDEF data
Gaia/apps/system	Target application is located in the directory, it expect NDEF messaged from various application like Browser, contact, Music, Video.
Nfcd	SnepServer protocol implementation, establish communication between the Low level driver and Gonk

V. Conclusion

This paper briefs the high level architecture of Firefox Mobile operating system, basic requirements of porting new Hardware platforms and NFC software libraries for Integration. This paper helps to illustrate NFC application development on the Firefox platform. Firefox is currently evolving mobile platform, will absolutely go for long run because of open source framework. Device manufacturer and network providers have pace with development on Firefox. Huge scope of application development in Mobile domain. Android running Hardware platform will ease migration to Firefox Operating system with minimal configuration changes.

References

[1] https://hacks.mozilla.org/2013/08/localizing-firefox-os-apps
[2] http://www.forbes.com/sites/quora/2013/05/13/what-are-the-major-changes-that-android-made-to-the-linux-kernel/
[3] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Gaia/Build_System_Primer/
[4] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Security
[5] https://developer.mozilla.org/en-US/docs/Mozilla/Gecko
[6] https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Platform/Architecture