Designing and Prototyping WebRTC and IMS Integration using Open Source Tools

Submitted in fulfilment of the

requirements for the degree of

Master of Science

of Rhodes University

by

Tebagano Valerie Motsumi

March 2018

Acknowledgements

I heard from someone that when you are pursuing a postgraduate degree, you become the product in addition to the degree itself. I can honestly say that these words ring true for me given how this research journey has stretched me in ways I never imagined possible. I learnt what it means to trust; to pursue relentlessly; to overcome and to be vulnerable enough to receive support and for that I thank Poppa God for being faithful and true to His word that indeed He will never leave me nor forsake me. A special thank you to my supervisors Dr. Mosiuoa Tsietsi and Prof. Alfredo Terzoli - Mos for the hands-on support he so selflessly gave me from beginning right up to the end and Alfredo for the oversight and financial support – words cannot express the immense gratitude I feel for you for your support throughout this journey. I want to thank my mum for her continued tender love and care during *our* project and for reminding me every time I faced challenges that I will always have a home. A big thank you to my sister Sega for her support, to my husband Sydney for his persistent encouragement to keep trucking and to my family and friends. If I have forgotten to thank you, please charge it to my head and not my heart. At last, it is finished.

Finally, thank you to my sponsors: Telkom SA, Coriant, Tellabs, Bright Ideas 39 and Easttel who generously provided the financial support that allowed me to complete this work.

Abstract

WebRTC, or Web Real-time Communications, is a collection of web standards that detail the mechanisms, architectures and protocols that work together to deliver real-time multimedia services to the web browser. It represents a significant shift from the historical approach of using browser plugins, which over time, have proven cumbersome and problematic. Furthermore, it adopts various Internet standards in areas such as identity management, peer-to-peer connectivity, data exchange and media encoding, to provide a system that is truly open and interoperable. Given that WebRTC enables the delivery of multimedia content to any Internet Protocol (IP)-enabled device capable of hosting a web browser, this technology could potentially be used and deployed over millions of smartphones, tablets and personal computers worldwide.

This service and device convergence remains an important goal of telecommunication network operators who seek to enable it through a converged network that is based on the IP Multimedia Subsystem (IMS). IMS is an IP-based subsystem that sits at the core of a modern telecommunication network and acts as the main routing substrate for media services and applications such as those that WebRTC realises. The combination of WebRTC and IMS represents an attractive coupling, and as such, a protracted investigation could help to answer important questions around the technical challenges that are involved in their integration, and the merits of various design alternatives that present themselves.

This thesis is the result of such an investigation and culminates in the presentation of a detailed architectural model that is validated with a prototypical implementation in an open source testbed. The model is built on six requirements which emerge from an analysis of the literature, including previous interventions in IMS networks and a key technical report on design alternatives. Furthermore, this thesis argues that the client architecture requires support for web-oriented signalling, identity and call handling techniques leading to a potential for IMS networks to natively support these techniques as operator networks continue to grow and develop. The proposed model advocates the use of SIP over WebSockets for signalling and DTLS-SRTP for media to enable one-to-one communication and can be extended through additional functions resulting in a modular architecture. The model was implemented using open source tools which were assembled to create an experimental network testbed, and tests were conducted demonstrating successful cross domain communications under various conditions. The thesis has a strong focus on enabling ordinary software developers to assemble a prototypical network such as the one that was assembled and aims to enable experimentation in application use cases for integrated environments.

Table of Contents

List of Figures

List of Tables

1. Chapter 1 – Introduction

1.1. Overview

Over the years, there has been a strong trend towards faster provisioning of real-time multimedia services over the Internet. In a recent annual report measuring the world's adoption of Information and Communications Technology (ICT), ITU (2016) state "that while 84 percent of the world's people live in an area where mobile-broadband services are offered, only 47 percent are actually using the Internet" (ITU, 2016). The accessibility that the Internet provides to users at a global scale enables the use of different devices over which users can utilise dynamic and tailored services to communicate anytime, anywhere. The growing trend for more efficient services has also led to improved requirements for network architectures where the Internet Protocol (IP) is the *de facto* standard. As such, IP-based infrastructure has been deployed resulting from a shift from circuit to packet-switched networking (Black, 2001). The IP Multimedia Subsystem (IMS) was thus developed to provide a common IP platform that facilitates multimedia service creation, deployment and supports interoperability between the Internet and legacy cellular systems (Khandelwal, 2007).

The Third-Generation Partnership Project (3GPP) is the recognised custodian of IMS standardisation which was initially intended to deliver new mobile services over evolved Universal Mobile Telecommunications System (UMTS) networks (3GPP, 2017). The European Telecommunications Standards Institute (ETSI) also standardises IMS as part of its definition of Next Generation Networks (NGNs) where the Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN) working group is responsible for its specification. Work done by TISPAN also includes the definition of other non-IMS subsystems such as the Network Attachment Subsystem (NASS) and the Resource Admission Control Subsystem (RACS) which function at the transport layer. The NASS acts as a Dynamic Host Configuration Protocol (DHCP) server by providing IP addresses and other configuration parameters; authentication; authorisation and location management while the RACS applies policy decisions when managing resources and controlling user access based on their profiles (Brouquet, 2008).

The IMS is part of 3G/4G standards which are constantly being developed and extended. The 3GPP IMS and ETSI IMS are thus standardised where joint focus is on application service development, radio access networks, network convergence and so on. Other standardisation bodies are also involved in IMS standardisation, for instance, the Internet Engineering Task Force (IETF) are responsible for specifying the communication protocols used in IMS such as the Session Initiation Protocol (SIP) which is mandated as the main signalling protocol. Furthermore, the Open Mobile Alliance (OMA) is also enlisted to provide third-party service capabilities such as Push to Talk over Cellular (PoCC) (Bertrand, 2007). All the partners define IP networks and therefore approach IMS from different vantage points. So, for the common parts of IMS standardisation, the 3GPP is responsible for maintaining a single set of specifications in order for the resultant architecture to capitalise on economies of scale and cost reductions, an initiative which as per ETSI (2007) is referred to as Common IMS. Therefore, for simplicity, the 3GPP IMS will be the main focus of this thesis.

1.1.1. Background of the IMS Architecture

The IMS architecture includes key functions that are responsible for routing, locating and addressing terminals that are connected in different ways to IMS to enable registration and access to services on the network (Camarillo & Garcia-Martin, 2007). Using IP as the main bearer, these functions are connected to form one administrative IMS domain where subscribers can also register from another network domain or geographic location through roaming facilities. Camarillo & Garcia-Martin (2007)

continue to define core IMS as comprising Call Session Control Functions (CSCFs) and a Home Subscriber Server (HSS). A CSCF is an essential function in the IMS which processes SIP signalling and comes in three different types: Proxy (P-CSCF), Interrogating (I-CSCF) and Serving (S-CSCF).

The P-CSCF is the entry point for all SIP requests between IMS and a user terminal. It forwards responses and requests appropriately in its capacity as a SIP proxy server. The I-CSCF is a SIP server that is the first point of entry for external requests. Its address is listed in the Domain Name System (DNS) records of the domain such that when a SIP server is looking for the next hop for a message, it obtains the address of an I-CSCF of the destination domain and forwards the message to that destination. The S-CSCF carries the central function of the signalling plane because, in addition to basic SIP server functionality, it acts as a SIP registrar, meaning that it manages bindings between a user's IP address, port number and transport protocol, and their SIP Address of Record (AoR), which can be expressed as sip:user@example.com.

Consequently, the S-CSCF is responsible for providing charging and billing information due to its role in service provision, triggering and maintenance, and hence interacts with other mediation systems. The HSS is the central repository for user-related information that is required to handle multimedia sessions. This information includes but is not limited to location information, user profile, security information (required for authentication and authorisation purposes) and the address of the S-CSCF that is serving the user. The HSS interacts with the I-CSCF and the S-CSCF using Diameter, a AAA (Authentication, Authorisation and Accounting) protocol.

In its expanded form, IMS includes other functions such as the Media Resource Functions (MRFs) and Public Switched Telephone Network (PSTN) gateways as shown by Figure 1-1. These functions are responsible for interfacing with other legacy networks to provide what is referred to as network convergence. The diagram below shows the P-CSCF acting as a point of contact for a user terminal located in a visited network and accessing services from their home network over a radio access network. The IMS also defines interfaces to Application Servers (ASs) which may be hosted either within the home network, visited network or in a third-party, non-operator network.

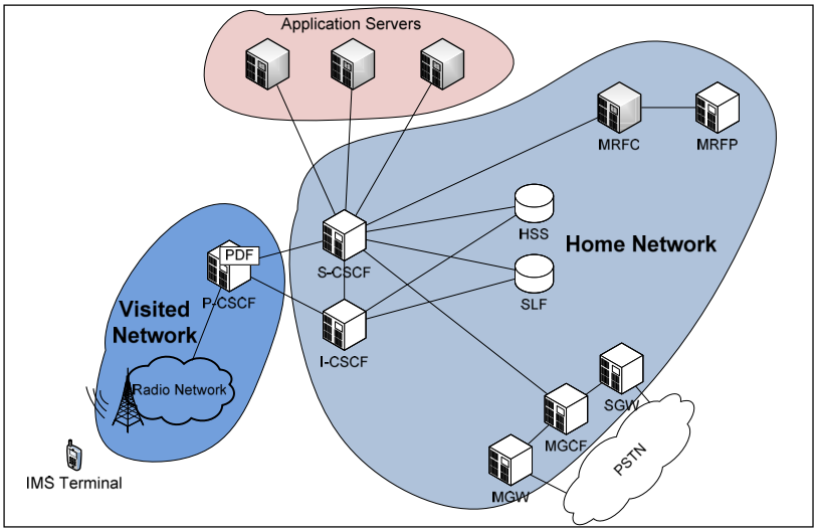

Figure 1-1 - The IMS architecture. Source: Brouquet (2008).

1.1.2. The Value Proposition of IMS

The value proposition of IMS as a service development platform is in its ability to provide Quality of Service (QoS), charging and the integration of different services as capabilities that can be

implemented universally for all applications. Such a service environment provides operators with the opportunity to differentiate their service offering through business models that can be tailored to meet user needs, where predictable communication experiences can be guaranteed at a premium, over and above the best-effort service level provided by the Internet. Camarillo & Garcia-Martin (2007) further describe the ability to apply different types of charging schemes that may be independent of the underlying business model adopted. With this in mind, the IMS was thus seen as the appropriate platform to position network operators as front-line competitors in service provision, where supporting multi-vendor products is a major objective. Moreover, Spiers & Ventura (2010) state further objectives of IMS as offering converged services to users over multiple access technologies; lowering costs associated with service creation including reducing the Time-To-Market (TTM) of services amongst others.

1.1.3. Poor Proliferation of IMS Services

In light of these objectives, however, there is a disconnect between the disruptive potential of IMS and the reality of the extent of its adoption, where IMS services are viewed as having not yet reached a subscriber base as large as originally envisioned by its custodians. According to Spiers & Ventura (2010), the steep learning curve associated with acquiring extensive knowledge of the multiple protocols, elements, interfaces and frameworks that constitute the IMS can have an impact on the ability to develop new and innovative services. This challenge is further met by the advent of web-oriented applications that, in addition to their innovative power, experience short development and TTM cycles that can easily be provided at relatively low costs. As such, the proliferation of IMS services is hampered especially when for every IMS service provided, there is a counter web application providing a similar, more enhanced service. Moreover, these web applications typically provide tailored service offerings, advertisements, user profiles etc. as a result of applying data analytics that allow a better understanding of the customer, hence optimising the user experience (Maes, 2010).

For example, social networking sites such as Facebook, Skype and Google Hangouts are already succeeding at changing the traditional view of telephony - call establishment between users requires their web identities as opposed to telco assets such as cell phone numbers or SIP addresses and telephony services (voice and video calls) are embedded within their communication suites and form only part of their overall service offering (Bertin, Crespi & L'Hostis, 2011). As a result of deploying their services 'over' existing telecommunication networks, these service providers have come to be described as over the top (OTT) service providers. Furthermore, OTT players are rewriting the web browser to become a common online platform for services with browser plug-ins as the fundamental technologies that have changed the way media is consumed on the web. By integrating telephony in these online platforms, which was a core service offering of the mobile operator, telcos are thus being relegated to the status of a mere 'data pipe' for other companies' data, as opposed to a major service provider (Raivio & Luukkainen, 2011).

1.1.4. The Open Telco Ecosystem

Maes (2010) suggests telcos use more open strategies to expand their ecosystem and offer more efficient and innovative services in order to compete against or collaborate with OTTs. Examples include offering their infrastructure as a service; enabling access to business resources such as QoS and Operating and Business Support Systems (OSS and BSS) and providing them as a service; providing a cloud computing platform to increase efficiency and reduce costs; becoming identity providers for web applications and federating these identities to curtail fragmentation, and so on. These examples indicate that the network operator may need to embrace web-oriented approaches. Raivio & Luukkainen (2011) suggest the adoption of hybrid strategies that are a balance between open and

closed ecosystems (or gardens) to create an Open Telco. Eisenmann, Parker & Alstyne (2008) define an open garden as a system that does not restrict how users interact with their products and services, and provide third-party integration through Application Programming Interfaces (APIs), while a closed system is one where the service provider restricts access to resources, application content and media. For example, Linux is an open system whereas Apple's iPhone is closed. Telecommunication networks are often seen as closed systems; therefore, the use of hybrid strategies would give them innovative power from opening up their networks via Open APIs while also maintaining control over their networks through business level agreements that govern the resultant strategic approaches.

The Open Telco ecosystem is a movement that was developed to improve telco agility with regard to transformation and innovation, and according to STL Partners (2015), using open source software would further enable their position as leading service providers. Their proposition is that open source software would result in technological progress; reduced financial pressure and increased agility when developing services with reduced TTMs. Spiers & Ventura (2010) also propose telcos release open source products that are created using widely accepted tools and programming languages which are easy to program or modify as a way to encourage developer interest in IMS. In light of the above, this thesis takes the pragmatic approach of presenting IMS integration with an open, standards-based web API known as Web Real-Time Communication (WebRTC).

1.1.5. The Emergence of Web-Based Standards

The emergence of web-based standards has been instrumental in growing the web ecosystem via native application development to meet increased demand from stakeholders for the adoption of alternatives to browser plugins. The amalgamation of the Hypertext Mark-up Language version 5 (HTML5), Cascading Style Sheets (CSS) and JavaScript as the main technologies behind web standardisation, in combination with JavaScript APIs, improves the accessibility and openness of the web (Amirante et al., 2013). These interfaces enable an application to use assets provided by a service provider, for instance, developers wishing to adopt a design like Facebook can use Facebook APIs as a set of building blocks over which this design consistency is to be maintained; resulting in the development of innovative applications that are available on any kind of supporting device, thus establishing what is known as the Open Web Platform (Eriksson & Hakansson, 2012). Many organisations are involved in the development of such a platform, particularly the Web Hypertext Application Technology Working Group (WHATWG) that have made significant advancements in the creation of APIs responsible for establishing real-time Peer-to-Peer (P2P) communications since 2006. This work progressed into a mature API standard that was ultimately taken up by a World Wide Web Consortium (W3C) Working Group in 2011 with the fundamental concepts of media access and the establishment of P2P connections thus being created.

The success of the Open Web Platform, in addition to web standardisation, also depends on advancements made within browser engines that are responsible for rendering advanced marked-up content and applying styles to such content (Eriksson & Hakansson, 2012). The four main browser engines implemented include WebKit, used in Safari and Google Chrome; Gecko, used in Mozilla Firefox; Trident, used in Internet Explorer and Presto used in Opera. These engines are also used in other less popular browser implementations, desktop applications and mobile devices running various operating systems. Of these, WebKit and Gecko are open source frameworks whereas the rest are closed source. Major technological developments are typically seen with open source engines because developers can publicly propose new features and implementations that are usually developed in parallel with on-going standardisation. These developments result in browser architectures that are more flexible, secure and cutting-edge than their proprietary counterparts, thus leading to major

strides within the web domain; as is evidenced by the proliferation of Google Chrome as the most widely used browser in the world as reported by StatCounter.com (2016).

1.1.6. The Advent of WebRTC

Google's acquisition of On2 and Global IP Solutions (GIPS) resulted in major developments within the field of real-time communication on the web. On2 developed the VP8 video codec and others in the VP (9 and 10) series, and GIPS developed media frameworks that provided support for JavaScript APIs to enable bi-directional media processing and coding technologies, particularly for use in Voice over IP (VoIP) systems (Alexandru, 2014). Consequently, Google released VP8 under a patent-free licence with the aim of providing a high-quality video codec to rival the widely used and patented H.264 video codec. Coupled with the W3C standardisation effort, an open source API implementation was released by Google as the platform over which developers could experiment with real-time communication on the web and was named WebRTC.

Interest in WebRTC includes companies such as Mozilla, Opera, Telenor, Cisco, Ericsson, Oracle and others, as well as private individuals. Apple, on the other hand, has been absent since the advent of WebRTC except for participating in the process of selecting an MTI video codec. The company has continually been adamant about support for H.264 with no interest in the VPx series. At the same time, WebKit, the rendering engine used by Safari, had an issue logged as far back as November 2013 for support for WebRTC. The status of this item remained as "in development" for several years, including the period of time during which much of this investigation was conducted. The status changed to "resolved" in August 2017 and as of March 2018, WebRTC is shown as "supported" (WebKit, 2018). WebRTC support was similarly taken up by standardisation bodies, resulting in the creation of working groups within the W3C and the IETF through which a consistent and stable standardisation process could be followed (Ubiquity, 2005). The W3C is tasked with defining the WebRTC API, while the IETF is tasked with defining the protocols and media processing mechanisms that extend the browser's RTC functionality (IETF, 2014; W3C, 2015). In addition to their involvement in working groups, these bodies continue to demonstrate the ability of WebRTC to support real-time capabilities on the browser through various "plug-and-play" applications. Figure 1-2 below shows a snapshot of browser support for WebRTC at the time of writing. The colour coding scheme shows the extent to which the API elements (left hand side) are supported by the major browser vendors (top) – red signifies that not all the API elements are supported; green signifies that most of the API elements are supported and yellow signifies that support for these elements is "in progress" or "under development".

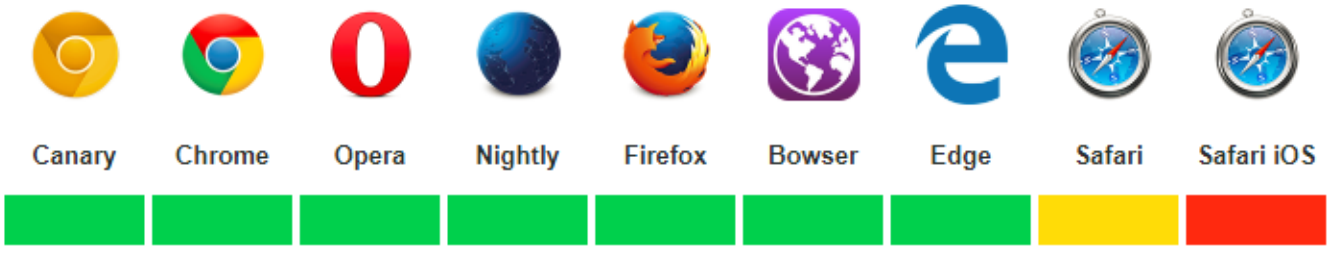

Figure 1-2- Browser support for WebRTC. Source: Talky (2018).

1.1.7. Discontinuation of Browser Plugins

WebRTC promises to add RTC capabilities to the web by providing browser support for direct communication with another browser without the need for third-party plugins such as Adobe Flash Player; Microsoft Silverlight; RealPlayer; QuickTime and many more that have played such a fundamental role in the deployment of rich and expressive multimedia content across different

browsers. Plugins have enabled browsers to act as multimedia platforms which have enabled user experiences that were otherwise not possible or feasible on the web (Davies, Zeiss & Gabner, 2012). Even though plugins have resulted in extensive modifications to browser architectures and are still widely used on the web, recent years have seen a shift away from their use. Early plugins were based on the Netscape Plugin Application Program Interface (NPAPI) architecture and major providers such as Google and Mozilla have discontinued support for NPAPI-based plugins such as Java and Silverlight, on their platforms (Chrome Help, 2015; Mozilla, 2015). Similarly, plugins such as Flash that are not based on the legacy NPAPI architecture provide a "*click-to-play*" feature to bypass the security restrictions implemented against plugins.

Furthermore, Apple discontinued support for plugins (particularly Flash) across their devices as stated by Jobs (2010) via an online public address. This reduced support was due to the wide criticism plugins received for the number of problems they presented to user when downloading, installing and updating, notably browser crashes; security vulnerabilities and code complexity. As browser architectures change over time - becoming faster, more secure and more capable - plugin vendors have battled to keep up-to-date with these changes, and even more so across multiple platforms, hence leading to the proliferation of web-based standards that support use cases previously implemented by plugins (Schuh, 2013).

1.1.8. WebRTC and IMS Integration

The progress of IMS standardisation is well developed with extensive documentation produced that describe various IMS interactions with other systems while contrarily, WebRTC is still an emerging web technology with standardisation that is still ongoing and currently under-developed. These differences present opportunities for operators to be frontline competitors in service provisioning, and can help them reach a critical mass number of users (Toutain, Le Huérou & Beaufils, 2015). According to Khandelwal (2007), IMS features such as QoS; session establishment; identity management and authentication; charging, billing, and more, act as value-added drivers that can be used to enrich WebRTC and IMS integrated services. Therefore, the telco adoption of WebRTC presents the potential to offer a wide range of use cases where WebRTC can also benefit from a stable and well-defined environment that IMS provides. These use cases would be facilitated by the platform or device independence of WebRTC where implementers need not be concerned with maintaining plugin software. Even though OTTs' adoption of WebRTC could disrupt the web, opportunities exist for the telco to also provide multimedia capabilities in different devices such as smartphones, tablets, netbooks, set-top boxes, TVs etc., where revenue generation has been a challenge for them (Raivio & Luukkainen, 2011).

The basic aims of standardisation therefore are: 1) to enable simple browser-to-X communication (where X is either another browser or a standard IMS terminal) and 2) to show support for integration with other systems. The scope of the thesis is to cover both of these aims, with a strong emphasis on the second point in reference to IMS. When inter-working with different networks and services, Bertin et al. (2013) give some examples of how companies such as Oracle, Cisco, Ericsson, Alcatel-Lucent and many more, have developed inter-working functions such as gateways, Session Border Controllers (SBCs) and application servers to interoperate with WebRTC. The availability of these intermediary functions has resulted in innovative business models and architectures supporting web-based signalling and media processing capabilities. In fact, many prominent open source projects such as

Asterisk[1], Kamailio[2], FreeSwitch[3], Kurento[4] and others already support WebRTC thereby allowing proof-of-concepts for standardisation initiatives such as those taken up by the 3GPP.

The 3GPP has recognised opportunities for network operators to expand their ecosystem and have expended much effort in investigating the possible architectural arrangements that could facilitate the delivery of WebRTC services over IMS (3GPP, 2013). They have thus drafted a Technical Report (TR) 23.701 that presents these architectural alternatives and expresses the high-level requirements for the network operator to fulfil when enabling their integration. It also describes the possible modifications to the IMS architecture to enable integration with WebRTC, hence representing the guiding framework behind how the IMS architecture can be reimagined in order to enable WebRTC access. Work done by Bach et al. (2014), Bertin et al. (2013) and Cruz & Barraca (2015) give some evidence of how TR 23.701 is used to guide the investigation of the integration scenario from multiple facets.

1.2. Research Problem

The complexity of the IMS environment requires a clear plan and a systematic approach in order to enable integration with a web-based standard such as WebRTC, and the 3GPP TR 23.701 provides a suitable basis for that. As previously stated by Spiers & Ventura (2010), the availability of easily programmable and extendable applications in the form of open source products has the potential to remove the barrier of entry for software developers wishing to experiment with such integrations. However, while TR 23.701 is a technical document that explores this topic, it is not presented in a suitable format to assist software developers in their experiments – rather the document is targeted at telecommunication engineers and network architects. As such, the report is presently inadequate for these audiences.

1.3. Research Goals

In light of the above, this thesis thus sets out to address the challenges that software and service developers would have to overcome to perform integrations with IMS. The proposed integration model may be realised over a prototypical, networked environment that can allow different kinds of developers with expertise in either web or telco environments to conduct further research and experimentation. The following main points summarise the research goals of the thesis:

- **To synthesise a WebRTC and IMS integration model that addresses developer needs and requirements.**

The purpose of this goal is to determine said developer needs and requirements by laying a theoretical foundation of the 3GPP groundwork and analysing the architectural solutions suggested in TR 23.701. This analysis ensures that the proposed model is practical and mimics, as far as possible, real-life networks that can be prototyped by developers.

- **To create an open source testbed that enables testing and experimentation.**

Realising this objective allows the research to respond to the challenge of how the proposed model can be prototyped so that developers can be able to test and experiment over it with the aim of evaluating its behavioural impacts within the RTC landscape. Segec & Kovacikova (2012) state that the

[1] http://www.asterisk.org
[2] http://www.kamailio.org
[3] http://freeswitch.org
[4] http://www.kurento.org

open source community is more competent compared to their proprietary counterparts at mapping standards to practice through products whose source code is freely available to the public for viewing, distribution, modification and redistribution. For this purpose, the resultant testbed aims to create a learning environment for users to grapple with WebRTC and IMS integration.

1.4. Research Objectives

For each goal, the objectives are listed below to detail the process the research will follow to investigate the important yet under-researched integration of WebRTC and IMS.

- **To synthesise a WebRTC and IMS integration model that addresses developer needs and requirements.**
    - To describe how the W3C and the IETF have envisioned WebRTC, listing the current open issues and subsequent efforts to overcome these challenges.
    - To present the IMS service architecture and bring forth the most prevalent requirements needed to integrate third-party services.
    - To critically analyse the solutions presented in the TR 23.701 and describe the set of functions and requirements that are necessary to enable WebRTC access to IMS.
    - To formulate a novel integration model that is based on these functions and requirements.

- **To create an open source testbed that enables testing and experimentation.**
    - To present the currently available open source software for implementing WebRTC and IMS testbeds/platforms.
    - To evaluate the suitability of these software tools and their ease of integration to enable developers to deploy WebRTC and IMS services.
    - To implement an initial prototype of the integration model using the selection of open source software.

1.5. Research Scope

The scope of this research is confined to the use case of enabling one-to-one communication between WebRTC and IMS endpoints. There are other use cases such as video conferencing, instant messaging, file transfer, live streaming and so on that can be supported, however, the objective of this restriction is to focus on the core issues that result from the emergent requirements of enabling the target use case. With this said, although the proposed model is designed to support multiple protocols and techniques, SIP over WebSockets is chosen as the main signalling protocol when illustrating the interactions between WebRTC and IMS, thus simplifying the construction of the solution architecture. The research focuses on using WebRTC in application servers to thoroughly investigate how the differences in the WebRTC and IMS systems can be resolved at the access edge.

The research is limited to using open source tools for experimentation as opposed to proprietary products -proprietary products could be compared with open source products to provide a more inclusive and comprehensive analysis, however, their use is infeasible given the time and financial constraints of the research.

1.6. Document Overview

The rest of the thesis is organised into seven chapters as follows:

1.6.1. Chapter 2 – The WebRTC Standard

This chapter describes how standardisation was taken up by the IETF and the W3C. The discussion also describes the WebRTC API and the protocols that are mandated to interact with it to adapt the browser as an engine for RTC. Furthermore, the chapter also describes the open issues faced during standardisation.

1.6.2. Chapter 3 – The IMS Service Architecture

This chapter focuses on the application servers that are part of the service architecture to describe how services are provisioned, including those provided by third-parties. The standardised interfaces implemented within this service architecture are also described, giving a synopsis of IMS requirements identified during service provision.

1.6.3. Chapter 4 – The Integration of IMS with WebRTC

This chapter provides an analysis of the architectural solutions presented in TR 23.701 and presents insights gained from this analysis. The reference architecture chosen by the 3GPP is also presented which was instrumental in laying a foundation to understand the different functional entities that enable the inter-connection between WebRTC and IMS. Moreover, requirements specific to the integration scenario are presented.

1.6.4. Chapter 5 – The Integration of WebRTC and IMS: Proposed Model

This chapter presents the suggested model that is informed by the solutions analysis conducted in the previous chapter. The 3GPP reference architecture proved valuable at structuring the presentation of the model where the design considerations behind the different entities are discussed, leading to a more practical architecture.

1.6.5. Chapter 6 –Prototyping the Model

The purpose of this chapter is to demonstrate the use of open source frameworks and their ability to create a network testbed that is extensible and modular, to allow further testing of many use case scenarios. This demonstration showcases the integration landscape based on what is currently done in practice.

1.6.6. Chapter 7 - Conclusion

The purpose of this chapter is to communicate how effective the solution was at satisfying research goals and objectives; and to place an emphasis on the main contributions of the research. Furthermore, the researchers' own analysis is provided which includes limitations and insights. Lastly, the possible extensions or use cases that could stem from the implementation are also discussed to present opportunities for future studies thereby situating the research strongly within the field.

2. Chapter 2 - The WebRTC Standard

2.1. Overview

This chapter details the standardisation efforts behind WebRTC and looks at how the different API components work together to enable a P2P connection. The WebRTC protocol stack is also described in order to detail how the IETF has envisioned the suite of communication protocols that work with the API. The chapter also gives an overview of the open technical, political and interoperability issues that are prevalent within this context, and concludes with a discussion on competing standards. This overview is important because it describes the key decisions that are being made around the WebRTC standard.

2.2. WebRTC Standardisation Efforts

The W3C and the IETF are the main standardisation bodies responsible for WebRTC. These organisations each have working groups that jointly develop the standard and comprise designers, researchers, vendors, developers, users and private individuals from the Internet community. The W3C's WebRTC Working Group, a subgroup of the Ubiquitous Web Applications Activity responsible for "enabling value-added services and business models for ubiquitous networked devices" (Hirsch & Braun, 2010), is tasked with defining the WebRTC API comprising a set of JavaScript APIs exposed to developers for application development (W3C, 2015). The IETF's RTCWeb Working Group is tasked with defining the protocols and media processing mechanisms that extend the browser's real-time communication functionality (IETF, 2014). The API is defined with the purpose of realising the use case requirements of simple video communication services, with the added responsibility of defining a multitude of innovative extensions as specified in Holmberg, Hakansson & Eriksson (2013).

2.3. The WebRTC Application Programming Interface (API)

The WebRTC API enables simple multimedia communication by providing the ability to access (capture) media streams from input peripherals such as cameras and microphones; to encode, decode and perform other forms of media stream processing such as echo cancellation; and to establish P2P connections employing Network Address Translation (NAT) and firewall traversal techniques where necessary. Overall, this functionality should successfully deliver media streams even in the presence of jitter and packet loss. The W3C WebRTC Working Group works in conjunction with the Media Capture Task Force, another W3C Working Group, to specify the API that enables access to local media devices (W3C, 2009). Furthermore, the lack of implicit trust of the web requires the implementation of security measures that assure them of strict privacy control where they are made aware of the type of media access and to where it is transmitted. Security considerations are addressed in collaboration with the IETF RTCWeb Working Group and are defined in an ancillary standard (Rescorla, 2015a, 2015b). The API specification involves the implementation of the functions that are illustrated in Figure 2-1 with further details provided in subsequent sections.

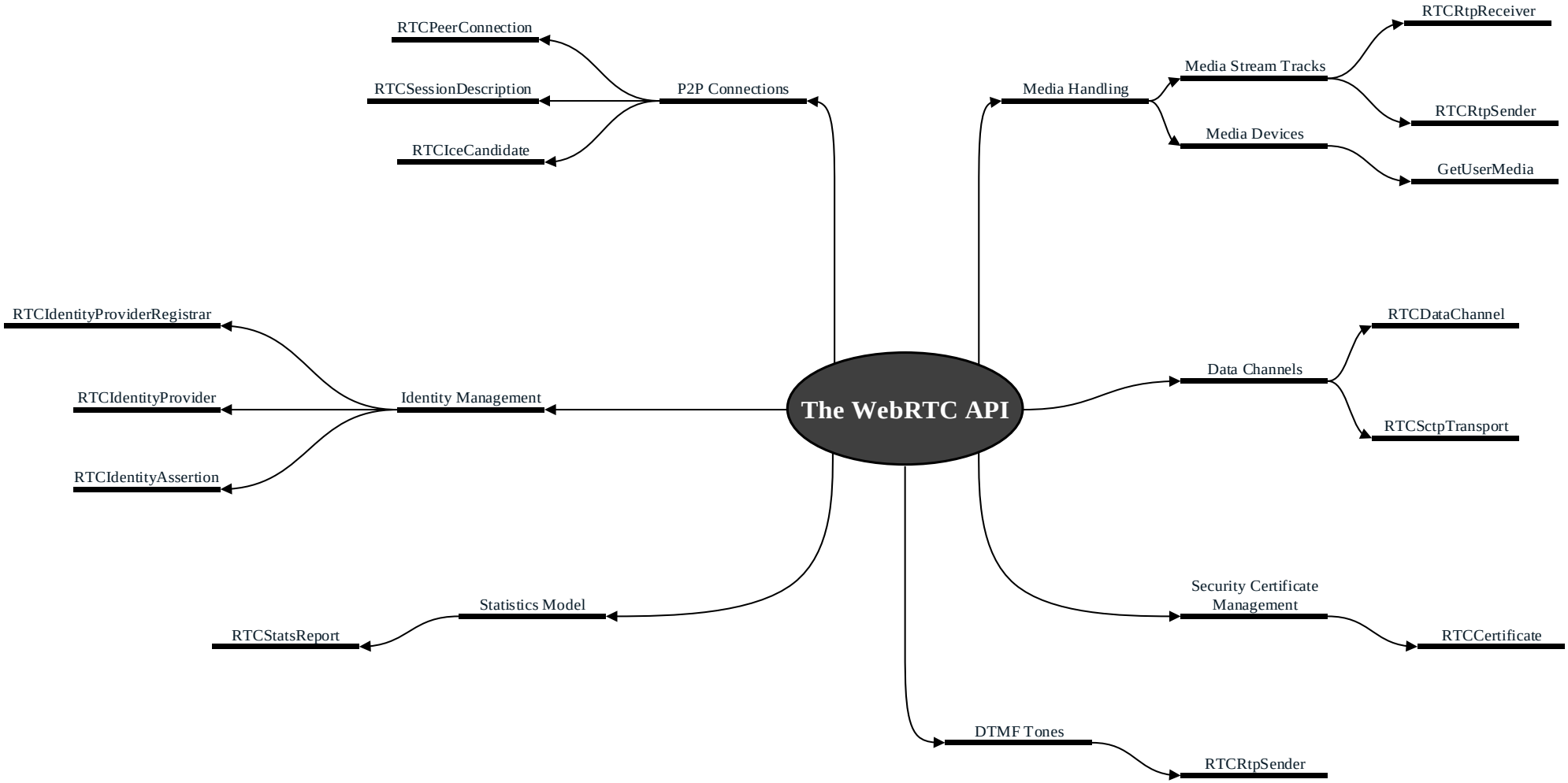

Figure 2-1 - The WebRTC API. Derived from W3C (2015).

2.3.1. Enabling Media Capture

The aim of the `RTPMedia` API is to define a `MediaStream` object that manages the generation, processing and rendering of media streams sent over a P2P connection. These media streams are captured from input devices via the `GetUserMedia` object. Media streams are represented as `MediaStreamTrack` objects that describe the type of media sent over the connection and comprise one or more audio and/or video streams. The `MediaStream` object also makes provision for remote access to media by specifying extensions for network use to enable varied types of media access. Figure 2-2 below illustrates this:

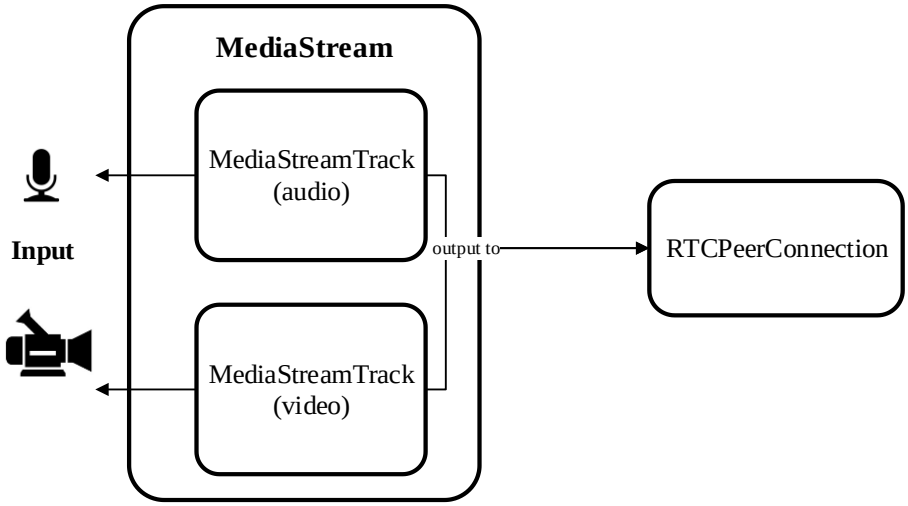

Figure 2-2 – WebRTC media representation. Adapted from Sredojev, Samardzija & Posarac (2015).

2.3.2. P2P Connections

The `RTCPeerConnection` object facilitates the negotiation of session description information, represented using the `RTCSessionDescription` object, which needs to be exchanged to establish a media path between browsers (the signalling channel used to coordinate this communication is not specified in WebRTC and is typically facilitated by a server function). The `RTCPeerConnection` object is also responsible for the exchange of arbitrary data, functionality that is handled by the `RTCDataChannel` object. Data channels are established in parallel to media streams without one causing congestion problems for the other.

2.3.3. Low Priority Functions

The WebRTC API also defines low priority functions that fulfil telephony-based requirements: the ability to send Dual-Tone Multi-Frequency (DTMF) tones during a call and the maintenance of a basic model to obtain statistics about the call session. The use of DTMF tones is to mimic classical telephony experiences, especially when inter-working with traditional fixed line terminals on the PSTN which support voice menu features to interact with services such as voice mail, airtime purchases, call centre queries etc. (W3C, 2015). The statistics model on the other hand, is implemented using the Real-time Transport Control Protocol (RTCP), a protocol that is responsible for monitoring and gathering statistics about a communication session. It can be used to implement QoS techniques to ensure reliable delivery and aids in synchronising multiple streams in conferencing scenarios where media is transmitted as separate RTP streams. Other low priority functions include the identity provisioning model which shows the interaction between the browser and an identity provider in order to authenticate the offers and answers exchanged.

2.4. Session Description in WebRTC

The W3C and the IETF do not define a precise mechanism as to how a browser should initiate communication with another browser on another machine. This was done by design to encourage vendor innovation - a website can either implement its own proprietary protocol or choose to use an existing protocol such as SIP or Jingle (Loreto & Romano, 2014). The standardisation bodies do however, define mechanisms to enable a WebRTC application to describe and negotiate media parameters to setup a session. These mechanisms include the conjunctive use of the Session Description Protocol (SDP) and a standard called the JavaScript Session Establishment Protocol (JSEP) (Uberti, Jennings & Rescorla, 2015).

SDP is used to describe media parameters and is the basis for the offer/answer model that enables endpoints to present and negotiate the desired media properties of a session (Rosenberg & Schulzrinne, 2002). Thus, the protocol describes various aspects of the session represented by different streams: audio; video; whiteboard; ICE transport information; security parameters and other media-related parameters. Having used SDP to describe a media session, JSEP is then used as a signalling abstraction layer where the exchange of offers and answers occurs via the `RTCPeerConnection` and with signalling messages defined in the format of the signalling protocol that the application has chosen to use (e.g. SIP or Jingle). In the case where SIP is used, the application would adapt the JSEP API into a SIP-compliant one where a `JSEP OFFER` is mapped to a `SIP INVITE` request, and say, a `200 OK` response to a `JSEP ANSWER` (Ravindran, Rauschenbach & Manickam, 2013).

2.4.1. JSEP Alternatives

Uberti et al. (2015) mentions other approaches to signalling that were considered as alternatives to the JSEP model. One approach considered the implementation of a lightweight signalling protocol (RTCWeb Offer/Answer Protocol (ROAP)) that imposes greater control over the generation and exchange of signalling messages on the web browser. The approach was rejected due to the complexity of having the browser maintain signalling state. Another approach included the ability to provide APIs to independently control media devices without having to generate session descriptions. Such an API definition was found to be cumbersome and would impede the WebRTC standardisation process given the need to arrange media interactions that would also need to be agreed upon and documented.

Further, Uberti et al. (2015) describe another approach that defines a `getCapabilities` interface where the application would have to generate session descriptions and subsequent offers and answers from the media capabilities of the media devices. This approach provides a further abstraction layer and thus adds a more complex set of interactions for the application to resolve. These approaches were considered based on their ability to ensure interoperability with other third-parties while creating a simple API platform for application developers from different backgrounds (either VoIP or web) to experiment with. Ultimately, the JSEP approach was chosen for its ability to provide a signalling solution within WebRTC where the application is given greater control with reduced complexity while specifying the generation of session descriptions in a manner that can be easily adapted to suit the underlying signalling protocol.

2.4.2. Example of Session Negotiation

Assuming the goal of simple one-to-one browser communication, the following scenario is used to describe the current session negotiation model of WebRTC within a single domain. The scenario illustrates the interactions between WebRTC API components employing JSEP to handle the exchange

of session description information. Alice and Bob are two hypothetical users who are both running the same WebRTC client, a JavaScript application downloaded from a web server that provides a user with access to communication services. This application, accessible over a WebRTC-enabled web browser or device capable of running JavaScript, enables a user to make and receive calls, instant messaging, file exchange, screen sharing, gaming and many more. To initiate the communication, both Alice and Bob have registered user accounts/profiles with the service provider and have each other's user identities in their contact lists. Alice clicks on a button to initiate a call with Bob and the application then instantiates an `RTCPeerConnection` object and makes an association with Bob's peer.

The `GetUserMedia` API adds a `MediaStream` object to an `RTCPeerConnection` and the media, transport and security parameters are generated using the `createOffer` method to set up a local configuration of Alice's media capabilities. The `setLocalDescription` method creates a local description in her peer while the `setRemoteDescription` creates Bob's remote description also at her peer. Alice's offer is sent to Bob via a WebSocket channel using a means not yet standardised. The signalling server is responsible for facilitating message exchange between Alice and Bob to determine the intended recipient of the message. The application alerts Bob of the call and upon answering, his remote peer processes the incoming message and instantiates an `RTCPeerConnection` object in response. Bob's browser follows a similar process to send back his own transport and security parameters via the signalling service, but uses the `createAnswer` method to generate these parameters. Figure 2-3 illustrates these interactions:

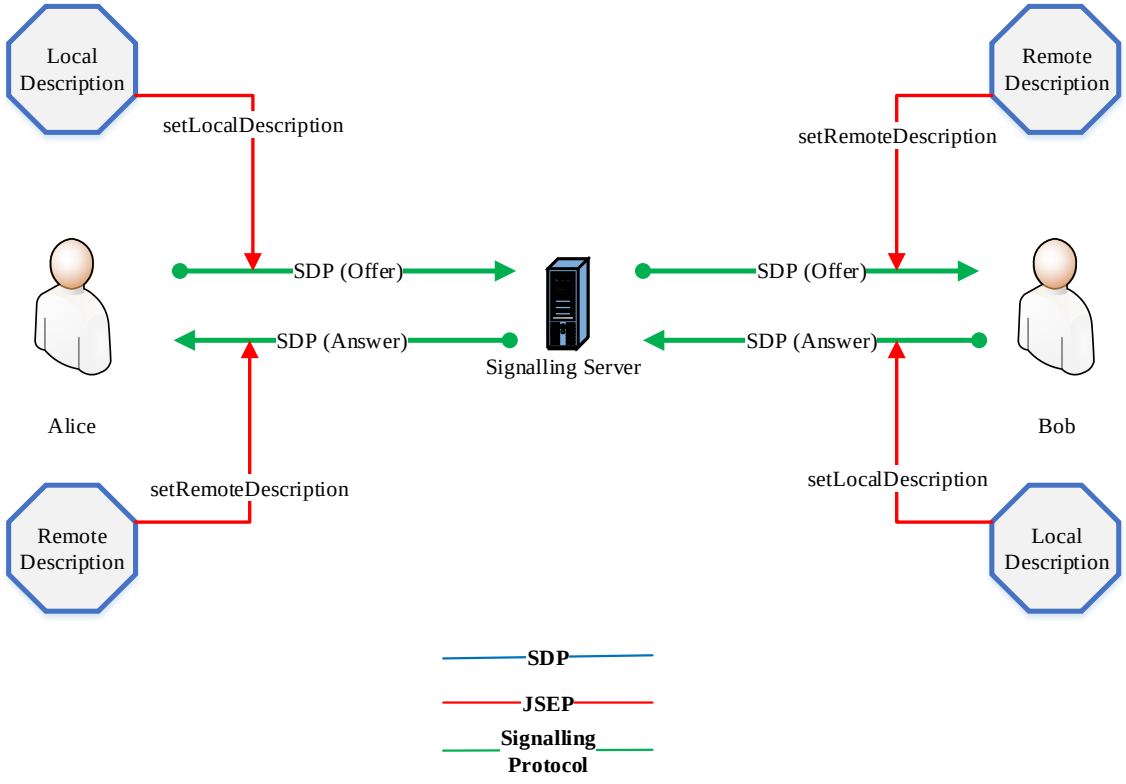

Figure 2-3 - Offer/answer model between Alice and Bob. Adapted from Sredojev et al. (2015).

2.5. The WebRTC Protocol Stack

The WebRTC protocol stack involves the use of Interactive Connectivity Establishment (ICE) to establish and maintain media connections, the Secure Real-time Transport Protocol (SRTP) to transmit audio and video securely, while the Stream Control Transmission Protocol (SCTP) transmits arbitrary

data. Media channels are encrypted using the Datagram Transport Layer Security (DTLS) protocol over the User Datagram Protocol (UDP) at the transport layer, whereas the WebSocket protocol functions at the application layer to enable the exchange of application control information. These protocols and the mechanisms invoking their use are discussed in the following sections.

2.5.1. Connection Management using ICE

The most basic WebRTC use case aims to enable P2P connections between endpoints residing in the same domain, where a media path is established without any intermediary firewalls or NATs. However, in the case where peers reside on different administrative domains, attempting to establish a P2P connection without any special handling would fail given the use of private addresses. To circumvent this connection failure, WebRTC endpoints implement ICE agents as a mandatory feature for negotiating the best communication path during NAT traversal (Janczukowicz, Bouabdallah & Bonnin, 2015). ICE works by compiling a list of IP addresses and ports in SDP offers and answers, then testing them for reachability using the Session Traversal Utilities for NAT (STUN) and Traversal Using Relays around NAT (TURN) protocols. STUN helps an endpoint to determine its address allocated by a NAT and therefore enables it to communicate with peers outside of its network. Moreover, it also helps the client determine the topology of the network in which it resides (for example the kind of NAT it is sitting behind) by sending requests to multiple STUN servers, while also providing a security measure against untrusted webpages and applications.

On the other hand, TURN relays packets between endpoints behind NATs and is usually implemented as a last resort because of the effectiveness of STUN at finding a public routing path. Furthermore, it needs higher bandwidth because of the need to relay multimedia through an intermediary (Mahy, Matthews & Rosenberg, 2010). The ICE agent is managed as a layer within the WebRTC framework and thus needs to interact with the `RTCPeerConnection` object to deliver ICE messages via the signalling channel (Eriksson & Hakansson, 2012). When establishing a media connection, each peer appends its own IP address and forwards it to the other via SDP strings exchanged during session establishment. The SDP string also contains the port numbers through which the media connection is to take place.

The ICE agent generates an ICE candidate which provides information about the IP address and port number of the server employed during connection management. Moreover, the ICE agent is also responsible for keeping the P2P connection alive. The agent is configured within an `RTCPeerConnection` object to listen to any ICE events that may occur during the candidate gathering process. An example of an ICE event includes STUN requests and responses that are exchanged between peers to ensure a consistent connection – this process serves as a connection keep-alive where TURN is used as a fall-back strategy in the event of a connection failure (Mozilla, 2016). Figure 2-4 illustrates the above process.

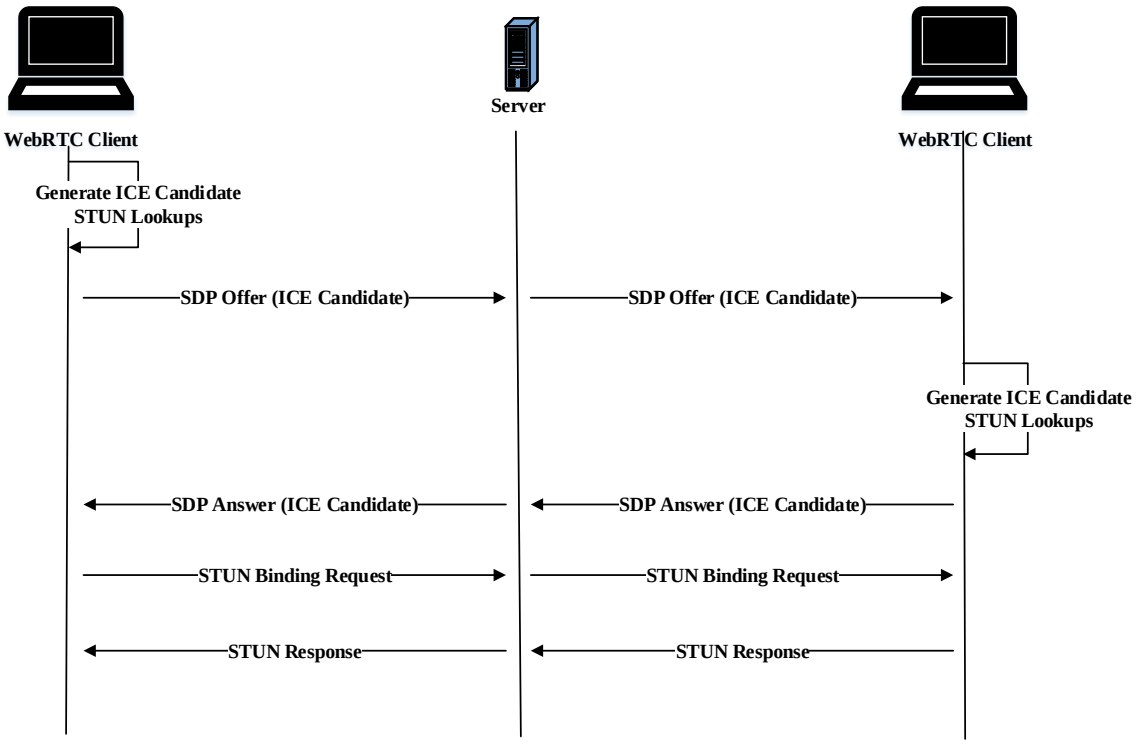

Figure 2-4 - Connection management using ICE. Adapted from Rosenberg (2010).

2.5.2. Audio and Video

The Real-time Transport Protocol (RTP) is used to transport real-time multimedia and is implemented by the media engine of a WebRTC-enabled component. RTP is an application layer protocol that typically runs over UDP to realise best-effort delivery of media packets and works in concert with other complementary protocols such as RTCP to monitor QoS. Other than providing a mere reporting function, RTP+RTCP do not guarantee in-order delivery of media packets, nor do they assume a reliable network connection. As such, according to Schulzrinne, Casner, Frederick & Jacobson (2003), RTP+RTCP support the basic transportation of interactive media by providing functionality that includes: “payload type identification, sequence numbering, timestamping and delivery monitoring” (Schulzrinne et al., 2003) in order to satisfy media requirements common to a wide number of applications. For those applications with additional requirements, RTP is by design, a protocol that is extensible and hence open to extensions through the definition of a new RTP profile: a specification that defines the payload type and format of media encodings. Figure 2-5 shows the payload information for standard audio and video encodings:

PT	Encoding Name	Audio/Video (A/V)	Clock Rate (Hz)	Channels	Reference
0	PCMU	A	8000	1	[RFC3551]
1	Reserved				
2	Reserved				
3	GSM	A	8000	1	[RFC3551]
4	G723	A	8000	1	[Vineet_Kumar][RFC3551]
5	DVI4	A	8000	1	[RFC3551]
6	DVI4	A	16000	1	[RFC3551]
7	LPC	A	8000	1	[RFC3551]
8	PCMA	A	8000	1	[RFC3551]
9	G722	A	8000	1	[RFC3551]
10	L16	A	44100	2	[RFC3551]
11	L16	A	44100	1	[RFC3551]
12	QCELP	A	8000	1	[RFC3551]
13	CN	A	8000	1	[RFC3389]
14	MPA	A	90000		[RFC3551][RFC2250]
15	G728	A	8000	1	[RFC3551]
16	DVI4	A	11025	1	[Joseph_Di_Pol]
17	DVI4	A	22050	1	[Joseph_Di_Pol]
18	G729	A	8000	1	[RFC3551]
19	Reserved	A			
20	Unassigned	A			
21	Unassigned	A			
22	Unassigned	A			
23	Unassigned	A			
24	Unassigned	V			
25	CelB	V	90000		[RFC2029]
26	JPEG	V	90000		[RFC2435]
27	Unassigned	V			
28	nv	V	90000		[RFC3551]
29	Unassigned	V			
30	Unassigned	V			
31	H261	V	90000		[RFC4587]
32	MPV	V	90000		[RFC2250]
33	MP2T	AV	90000		[RFC2250]
34	H263	V	90000		[Chunrong_Zhu]
35-71	Unassigned	?			
72-76	Reserved for RTCP conflict avoidance				[RFC3551]
77-95	Unassigned	?			
96-127	dynamic	?			[RFC3551]

Figure 2-5- Payload information for standard audio and video encodings. Source: Casner (2016).

2.5.3. Data

The exchange of arbitrary data between browsers is enabled by an association between the Data Channel Establishment Protocol (DCEP) and the Stream Control Transport Protocol (SCTP) which together provide reliable and ordered data transport (Jesup, Loreto & Tuexen 2015a, 2015b). DCEP establishes a bidirectional data channel by associating two unidirectional channels and setting the incoming or opening handshake streams to have the same stream identifiers. The use of SCTP then provides a way to encapsulate DCEP streams into an association over which certain requirements need to be met by endpoints. These requirements include the ability to establish data channels parallel to the SRTP media streams created by the `MediaStream` API over an `RTCPeerConnection` object; provide reliable, semi-reliable and unreliable data channels to support a wide variety of use cases from instant messaging to real-time gaming, and to enable congestion control. Furthermore, the use of message fragmentation, where large messages are sent without delaying data transmission via other data channels, is supported with the ability to provide efficient sequencing capabilities for in-order or out-of-order message delivery. The SCTP association is then encrypted over DTLS to provide confidentiality, source authentication and integrity protection for SCTP packets.

2.5.4. The WebSocket Protocol

When a user interacts with a web server to access communication services, the browser sends a request to the web server for the content, to which the server responds with the information requested within a standard Hyper-Text Transfer Protocol (HTTP) response object. This interaction describes the traditional setting where the server does not need to independently send data to the client without first having received a request. Recent trends now require the server to send data to the client on an as-needed basis. As such, current HTTP bidirectional technologies employ different techniques that interrogate the server with frequent requests for updates without a user having to constantly refresh a web page – these techniques are termed HTTP polling and require the server to delay responding to a client request until new data is available (Pimentel & Nickerson, 2012). According to Skvorc, Horvat & Srbljic (2014), the main issue experienced with such polling techniques is that the server must maintain multiple connections for each client request, a computationally and spatially expensive process that requires the maintenance of bindings between incoming and outgoing connections.

For this purpose, WebRTC mandates the use of another communication protocol namely, the WebSocket protocol, to establish a two-way, bidirectional communication channel between a client and a server that operates through a single socket. The protocol was defined to provide a reliable and suitable alternative to HTTP polling, hence it is based on HTTP and is designed to reuse existing HTTP server infrastructure for proxying, filtering and authenticating client requests. As such, it uses the standard HTTP port 80, and for secure connections, port 443. The protocol has two parts, a handshake and data transfer. The handshake is based on HTTP and begins when a client sends an HTTP GET method with an "*Upgrade*" request to the web server and upon successful negotiation, data transfer occurs where a bidirectional communication channel is established through which each side can independently send data.

The use of WebSockets for signalling is a growing phenomenon where signalling messages are exchanged by endpoints and is predicated on the client and server both agreeing on a protocol to use over the WebSocket connection. Examples of commonly used application protocols are SIP, JavaScript Object Notation (JSON) and certain proprietary protocols. The success of WebSockets is based on its ability to create scalable, real-time applications that place less burden on servers due to the reduction in network traffic and latency, hence easier management of multiple concurrent connections (Fette & Melnikov, 2011). Table 1 shows an example of a WebSocket handshake initiated by the client and responded to by the server:

The WebSocket Client Request

`GET /chat HTTP/1.1`
`Host: server.example.com`
`Upgrade: websocket`
`Connection: Upgrade`
`Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==`
`Origin: http://example.com`
`Sec-WebSocket-Protocol: chat, superchat`
`Sec-WebSocket-Version: 13`

The WebSocket Server Response

`HTTP/1.1 101 Switching Protocols`
`Upgrade: websocket`
`Connection: Upgrade`

```
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Table 1 - Example of a WebSocket handshake. Source: Skvorc et al. (2014).

2.6. Basic WebRTC Architecture

Figure 2-6 below sums up the overall WebRTC API and protocol stack interactions as implemented by WebRTC-enabled devices such as web browsers and gateways.

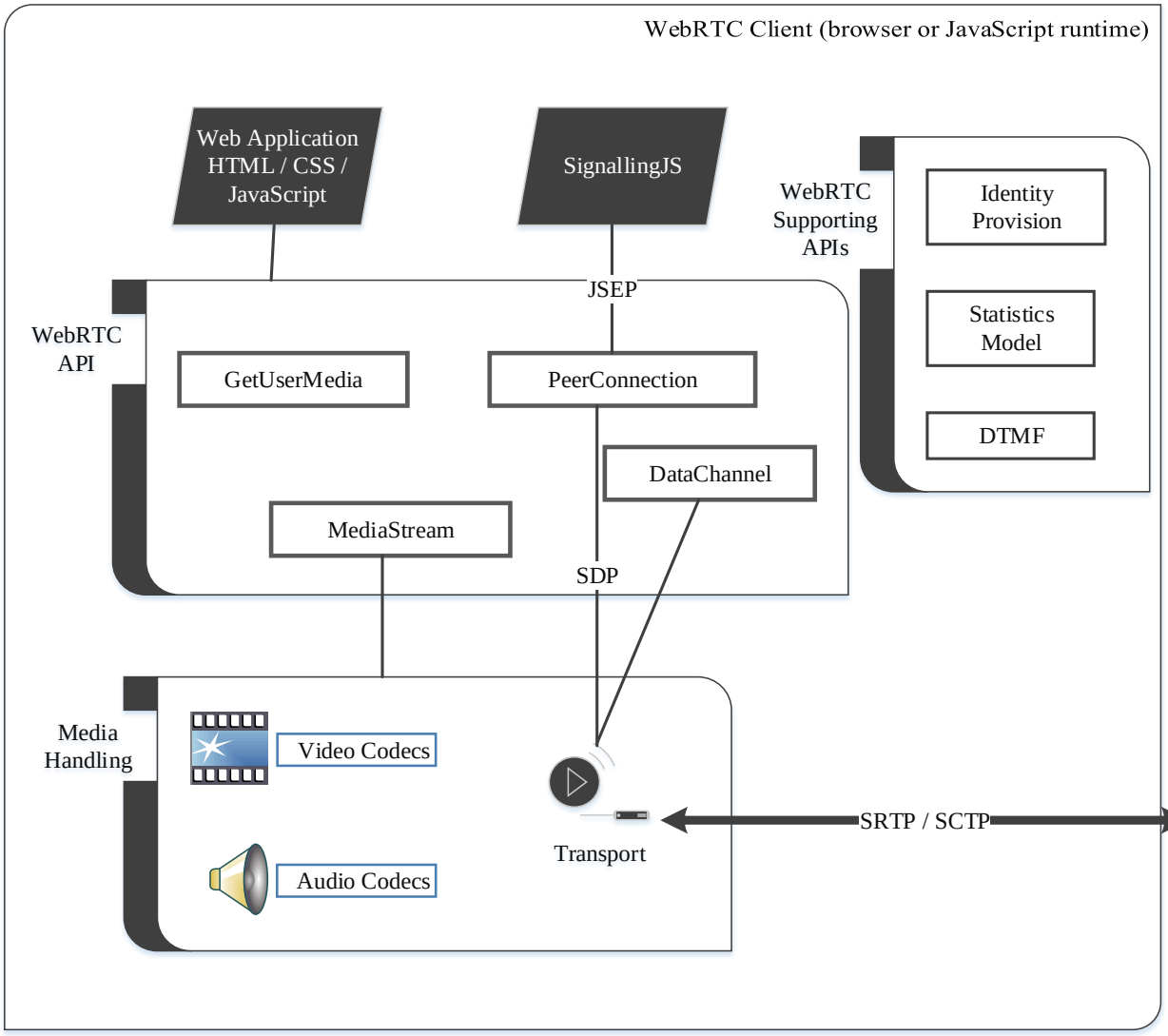

Figure 2-6 - WebRTC architecture and protocol stack. Adapted from Johnston & Burnett (2013).

Primarily, WebRTC is a P2P communication technology but it employs server intervention mainly on the signalling plane to enable session negotiation and connection management and ensure that media capabilities are exchanged by endpoints that could be located behind NATs or firewalls. JSEP, with media capabilities described by SDP, enables this negotiation upon interaction with a signalling protocol whose messages are transported using WebSockets. The web server is the main functional entity employed during this process to proxy signalling messages to endpoints participating in the communication session. Once established, the media path is setup where voice, video and arbitrary data are transmitted in a P2P fashion via the `RTCPeerConnection` object implemented by the browser engine. If some private network places restrictions on one or more peers, a STUN server finds

the best communication path for media with the TURN server used as a last resort for media relay. Figure 2-6 therefore illustrates a basic WebRTC model that should be supported as per the W3C and IETF standards - functionality beyond this basic structure is taken up by other standardisation efforts and defines the facilities to provide especially when enabling WebRTC support onto other web or VoIP-based systems.

2.7. Open Issues in the WebRTC Deployment Environment

The analysis of WebRTC which was conducted as part of this thesis highlighted several open technical, political and interoperability issues that continue to cause controversy within the WebRTC community due to the way different schools of thought have sought to assert their influence on the standardisation process. Some believe that ambiguity fosters room for innovation while others see it as having the potential to negatively impact the proliferation of the WebRTC standard as a whole (York, 2013). These open issues include how to approach identity provision and management; the definition of a Mandatory-To-Implement (MTI) video codec; the definition of standard signalling protocols and others that are out of scope of this discussion: how to approach video conferencing, presence, address book integration and notifications.

2.7.1. Identity Provision and Management

The IETF RTCWeb Working Group proposes an identity model where a third-party Identity Provider (IdP) and the WebRTC service provider are completely decoupled during identity assertion (Muranyi & Kotuliak, 2013). The model relies on the IdP alone to assert the user's identity, while the service provider merely forwards assertions between users participating in a communication session. Decoupling the interaction between the service provider and the IdP implies a lack of trust of the service provider while the IdP is wholly trusted to provide identities. Beltran et al. (2014) state that the service provider should be allowed to manage user identities because they are involved in the delivery of messages between users, and as such, are often better suited to manage a user's identity because it enables them to monitor their activity and provide tailored services. On the contrary, the WebRTC identity model restricts identity management by the service provider in order to avoid the potential scenario where the service provider unethically accesses user information without their consent. As a possible intervention, the user should have the liberty on a case by case basis to indicate whether to trust the service provider or not (Rescorla, 2015a).

Put another way, the WebRTC model mandates that the called party bear the responsibility of asserting the calling party's identity with their IdP as opposed to the service provider bearing this responsibility. Therefore, during session negotiation, the calling user's identity is included as a session description parameter allowing the cryptographic construction of a unique call fingerprint that is also asserted when a media channel is established. In this way, call participants can trust that they are talking to the same user they communicated with during session negotiation. Figure 2-7 illustrates the proposed WebRTC identity model where Alice and Bob are two hypothetical users that verify each other's identity with the other's respective IdP. The diagram also illustrates a functional entity instantiated by the browser called the IdP Proxy that obtains and verifies Alice and Bob's identity assertions that are then forwarded by the web server operated by a service provider.

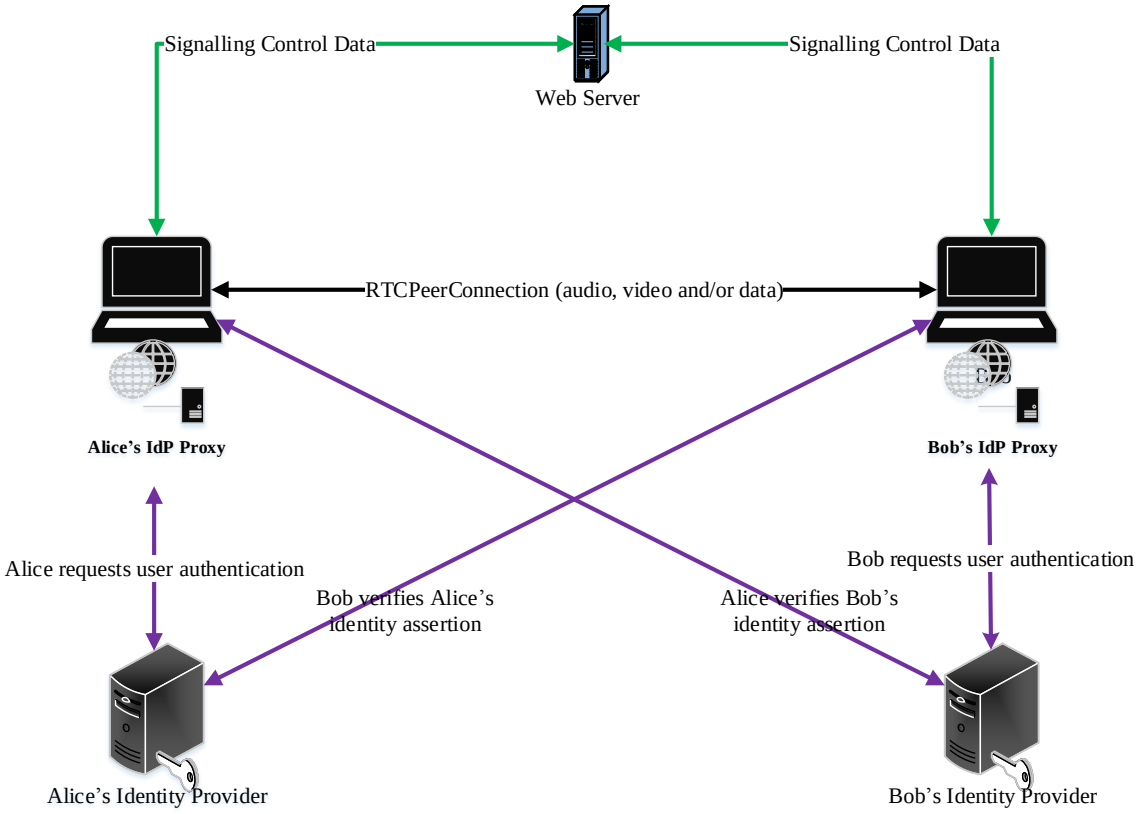

Figure 2-7 - The WebRTC identity model. Source: Loreto & Romano (2014).

In the diagram, Alice and Bob's browsers are omitted on purpose for simplicity (their interactions are mediated by the servers). The interaction is handled by a protocol-independent browser API that is currently, at the time of writing, still under discussion (Rescorla, 2015b). The purpose of the IdP Proxy is to "decouple the browser from any particular IdP" (Rescorla, 2015b) that way, the browser is able to handle multiple user identities implemented using different identity protocols, thereby acting as a bridge between the multiple services that the user is subscribed to, and the different IdPs providing a particular service. That is, when a service provider requests a user's identity, the IdP Proxy analyses the request and forwards it to the appropriate IdP. Thus, this approach creates a more efficient, secure and flexible identity system where users participating in a session are assured that they are communicating with the same party with whom session negotiation was initially conducted (Beltran et al., 2014).

2.7.2. Mandatory-to-Implement (MTI) Codecs

The vision for WebRTC to align with open source software is largely dependent on the licencing status of the underlying artifacts, in this case, codecs that collectively enable the successful implementation of WebRTC as an open web standard. The adoption of an MTI codec depends on that codec's ability to produce high quality media with good performance, and in such a way that the codec can be supported on a wide variety of hardware and software platforms. Furthermore, the Intellectual Property Rights (IPR) status of that codec also has a role to play in such discussions, hence the choice of an MTI codec, particularly video, has proven controversial, particularly regarding patents. These debates also stem from the way software distributors and hardware manufactures define the use of their intellectual property through different licensing structures that tend to promote their own codec and thus assert their commercial interests.

The IETF has defined a minimum requirement of Opus and G.711 as MTI audio codecs while at the time of writing the selection of MTI video codecs is under debate - the major contenders are VP8 and H.264/AVC constrained baseline profile – while the use of additional codecs is at the will of the implementer. Examples of supported audio codecs include iLBC, iSAC and for integration with VoIP systems: the Adaptive Multi-Rate (AMR) and the Adaptive Multi-Rate WideBand (AMR-WB) codec, G.722, Speex and others. In addition, recent variants of the current MTI video codecs, that is, VP9, VP10 and H.265, are also to be supported (Levent-Levi, 2014). In light of this codec selection, the main standardisation goal is to position WebRTC as a royalty-free project that has minimal restrictions on the use and distribution of its software. As a result, the Berkeley Software Distribution (BSD) licence protects WebRTC, and also enables the freedom around its use.

2.7.2.1. Audio Codecs

The consensus for the MTI audio codec seems to centre around Opus, followed by G.711 as the second choice. The ability for Opus to scale from low bitrate narrowband to full-band, ranging from 6 kbps to 510 kbps, makes it a high-quality option that is appropriate for a wide variety of audio applications including conferencing, in-game chat, live distributed audio performances and many more (Proust et al., 2015). This varied use of Opus also asserts its adaptability to changing network conditions, thus allowing sampling of audio at various effective rates (Table 2 illustrates this). At the same time, G.711 accommodates the integration of WebRTC with legacy VoIP-based systems due to its wide adoption on these systems. Even though the WebRTC community is contrary to this adoption, codecs within the G.7xx series are widely accepted within the VoIP community and have proven very adaptable (Narbutt & Davis, 2005). Ultimately, these codecs have a royalty-free status and provide high quality audio conversion, hence their adoption.

Abbreviation	Audio Bandwidth	Sample Rate (Effective)
NB (narrowband)	4 kHz	8 kHz
MB (medium-band)	6 kHz	12 kHz
WB (wideband)	8 kHz	16 kHz
SWB (super-wideband)	12 kHz	24 kHz
FB (fullband)	20 kHz (*)	48 kHz

Table 2 - Overview of Opus performance. Source: Narbutt & Davis (2005).

2.7.2.2. Video Codecs

Alvestrand & Grange (2013) describe VP8 as a royalty-free video codec typically used in a WebM media container that claims to encode and decode video with better performance than its counterpart H.264. Services such as Skype, Google Hangouts, and Firefox and so on, implement VP8. In addition, Google is continually licensing VP8 hardware accelerators to numerous chip manufacturers whose increasing support may enable the growth and success of VP8 as an MTI video codec (video processing is more expensive than audio and therefore often implemented in hardware that is not easily upgradeable). H.264 on the other hand has become the de-facto video standard in VoIP systems, is also supported by Skype, and so too has reached wide adoption across major web browsers. Proponents of each codec both claim that their supported codec outperforms the other based on

independent tests comparing the quality of conversion, thus resulting in an apparent hiatus in standardisation of the MTI video codec.

For WebRTC to reach mass adoption, proponents argue that standards need to mandate H.264 to avoid the risk of bypassing major adopters of RTC on the Internet while on the other hand; VP8 is reaching rapid adoption in real-time web applications and hardware devices. Nonetheless, the conclusion of the discussion around the MTI video codecs should result in a codec that produces a high video quality and performance; reasonable power consumption of both hardware and software implementations as well as a stable IPR status that will ultimately promote WebRTC within an open and competitive communication landscape (Bankoski, Wilkins & Xu, 2011). Currently, implementations adopt both codecs as a minimum requirement (Cardoza, 2015; Roach & Mozilla, 2015).

2.7.3. Signalling Alternatives

The ambiguous signalling landscape of WebRTC involves the use of SIP and JSON as the two most common protocols; others include Jingle, Open Peer or other proprietary solutions. SIP is a simple, extensible, flexible and familiar protocol that implements strong RTC capabilities to support a wide range of voice, video, data, file transfer, presence, instant messaging and other types of applications (Sege et al., 2014). Thus, it has played a major role in network convergence by facilitating the seamless integration of telco services over an interactive platform conducive to service creation and deployment, in the form of IMS. JSON on the other hand, is a language-independent text format that exchanges structured data between applications. It does this in a lightweight manner in the form of formatting rules applied to objects and/or arrays that make up JSON-text. The independence of JSON results in an instrument that supports a wide range of uses on the web, including the ability to send signalling messages between endpoints (Crockford, 2006).

Jingle is a key technology in the eXtensible Messaging and Presence Protocol (XMPP) that enables session establishment features comparable to SIP (Ludwig et al., 2016). As with SIP, Jingle session negotiation occurs over a signalling channel and the media is exchanged over a separate data channel. According to Jitsi (2011), inter-working functions between XMPP and SIP networks are feasible. Open Peer is a proprietary P2P signalling protocol developed by Hookflash which has been designed to support WebRTC services on the web browser, while also providing an independent signalling stack for standalone RTC applications (Raymond, 2012). The protocol addresses the scalability issues that are prevalent within SIP and XMPP while also bringing a novel signalling solution that covers a wide range of complex application scenarios.

2.7.4. Security Key Management Alternatives

WebRTC promotes inherently secure media transfer through the extended secure RTP profile. This secure profile ensures that WebRTC conforms to the widely accepted Confidentiality, Integrity and Availability (CIA) model for securing information systems by providing confidentiality through media encryption, integrity protection through message authentication and replay protection, and using cryptographic means to prevent unauthorised access to media (Kurose & Ross, 2012). The Datagram Transport Layer Security (DTLS) has also been specified for use with SRTP where DTLS handles the exchange of security parameters, algorithms as well as the derivation of secret master keys using during the media encryption process (McGrew, 2010). According to Pascual (2013), DTLS-SRTP was preferred over key management alternatives such as the SDP Security Descriptions for Media Streams (SDES); Multimedia Key Exchange (MIKEY) and ZRTP.

The wide adoption of SDES in commercial VoIP solutions has led to debates around the key management algorithm to use. Pascual (2013) states that the arguments made against SDES can also apply to DTLS. For instance, both eavesdropping and the insertion in the media path of a malicious web server are possible in both protocols. Furthermore, they are both susceptible to similar kinds of downgrade attacks. Kaplan (2015) also shares this view. However, DTLS-SRTP is able to secure signalling and media planes hence guaranteeing both their security unlike SDES and MIKEY which, as Pascual (2013) states, secures only the signalling plane via SDP independently of the media plane, thus proving insufficient when considering the security needs of the web. Proponents of SDES argue that it would provide backward compatibility when interworking WebRTC with VoIP networks and in response, DTLS-SRTP-to-SDES conversion should take place to enable interoperability with legacy VoIP networks. Still more, ZRTP was also considered because it "could potentially provide a simpler approach or even better protection in some scenarios" (Pascual, 2013).

In light of the above, a single mechanism, DTLS-SRTP, was chosen from an interoperability viewpoint thus when presented with key management options, an endpoint or gateway must select it above the others. Even though standardisation prohibits SDES use for WebRTC, industry implementations still provide its support and browsers such as Chrome serve as an exemplar, while Firefox only supports DTLS-SRTP. The vendor can thus determine the key management mechanism to adopt and in the same way as Adobe Flash; may support SDES and others via a flag or option that can be enabled at the developer's discretion.

2.8. Competing Standards

Currently, the latest versions of Google Chrome, Firefox and Opera support most of the WebRTC API components. Internet Explorer (now known as Edge) initially adopted a competing standard, Customizable, Ubiquitous Real-Time Communication over the Web (CU-RTC-WEB) which as stipulated by Bertin et al. (2013) later became the Object API for RTC (ORTC) until the first quarter of 2016 when WebRTC support also began to be included. ORTC implements session negotiation differently from the SIP-based offer/answer model accompanied by SDP, and provides web-oriented conditions that are more suitable for session negotiation to occur. An application can either send a multimedia track for voice or video, or set up a data channel to transfer other arbitrary data formats, as with WebRTC. Differences come about when implementing "sender", "receiver" and "transport" objects which are used to define "parameters", configured to describe what an object does and "capabilities", configured to describe the media, ICE and transport capabilities possible. The "sender" object bundles these configurations and transmits them to the "receiver" object for processing (Microsoft Developers, 2016). Figure 2-8 shows the interaction between the different objects to exchange media and data.

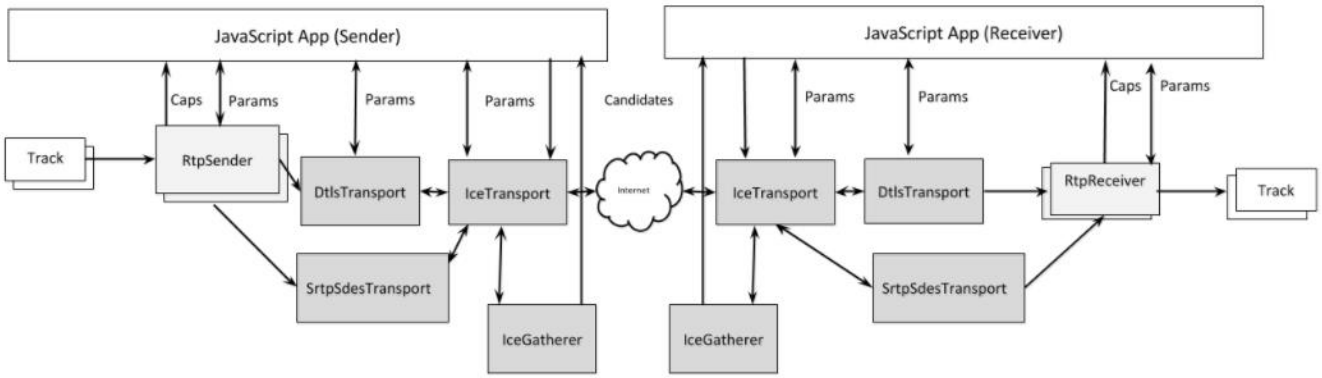

Figure 2-8 - ORTC object interactions. Source: Microsoft Developers (2016).

According to Cardoza (2015), ORTC in itself is not meant to replace WebRTC, even though some members of the RTC community consider it a likely successor of WebRTC (often called WebRTC 1.1), it is typically implemented as a layer on top of WebRTC which is used to extend its SDP functionality with support for backwards compatibility. Hence, both standards implement interrelated APIs, mainly the `RTCPeerConnection` API, thereby enabling interoperability between the two. Similarly, WebRTC implements some ORTC concepts such as the use of `RtpSender` and `RtpReceiver` objects. The signalling model for the ORTC approach is not clear because its definition came about mainly to address the challenges facing WebRTC regarding SDP functionality. The ORTC API can implement advanced capabilities such as layered video coding, simulcast, scalable video coding etc. more easily than WebRTC, which would require changes to the browser source code. The level of abstraction ORTC offers enables greater flexibility of the types of applications developed.

The Google WebRTC Project envisions a full convergence of WebRTC and ORTC where developers have the option to "upgrade" WebRTC to higher-level ORTC APIs with the freedom to bypass use of SDP (Microsoft Developers, 2016). This goes on to show that the rapid evolution of the web towards a standards-based environment is more reason for network operators to leverage WebRTC which is the channel through which RTC capabilities are added to the web browser.

2.9. Conclusion

This chapter provides an analysis of how the W3C and IETF are championing the standardisation process to define the interaction between the set of WebRTC APIs with the underlying communication protocols to provide native support for real-time communications capabilities on the web browser. This analysis served to show how the web platform has evolved over the years to support more dynamic and interactive content. The basic architectural model shown in *Section 2.6* is segue for subsequent chapters which look at adjusting this web model to enable WebRTC access to the IMS platform. The discussion of the controversial issues facing WebRTC expresses the challenges inherent in the standardisation process, where major stakeholders have the influence and commercial power to govern and augment its adoption. However, there are opportunities for these issues: identity provision; selection of MTI video codecs; signalling alternatives and security mechanisms, to be formally addressed in WebRTC's integration with IMS. As such, Chapter 3 covers the basics of IMS architecture, but more importantly seeks to emphasise the aspects of IMS that position it as a central integration platform for third-party services. The discussion will show how network operators historically have sought to evolve their networks, particularly in the area of multimedia services, with recent trends indicating an openness toward web-based paradigms.

3. Chapter 3 – The IMS Service Architecture

3.1. Overview

The purpose of this chapter is to give historical context to the IMS service layer which delivers multimedia services to subscriber using entities that are either resident in the home network or accessible via gateways into external domains. The web-based version of this integration channel using WebRTC necessitates an investigation into this context to determine the suitability of the IMS service platform to support an external system such as the one envisioned.

3.2. IMS Application Servers

The IMS service layer comprises ASs whose main purpose is to "host service containers in which applications are deployed" (Tsietsi, Honye & Thinyane, 2015). There are three types, each with a different approach to providing such services: the SIP Application Server; the Open Services Architecture (OSA) Service Capability Server (SCS) and the IP Multimedia Service Switching Function (SSF) (Bertin, Yahia & Crespi, 2007). Using these servers, it is possible to extend the IMS ecosystem to connect third-party services that can be deployed to various types of terminals without needing to change the endpoints themselves. As such, the service layer facilitates a coordinated service deployment strategy that benefits from the underlying functionality provided by the main application layer routing (CSCFs) and database (HSS) functions to deliver QoS, security, charging, billing and other enablers horizontally across services. Figure 3-1 illustrates the arrangement of these ASs in relation to each other and demonstrates how they interface with the S-CSCF as the main call control function. Subsequent sections describe their functions and roles during service execution.

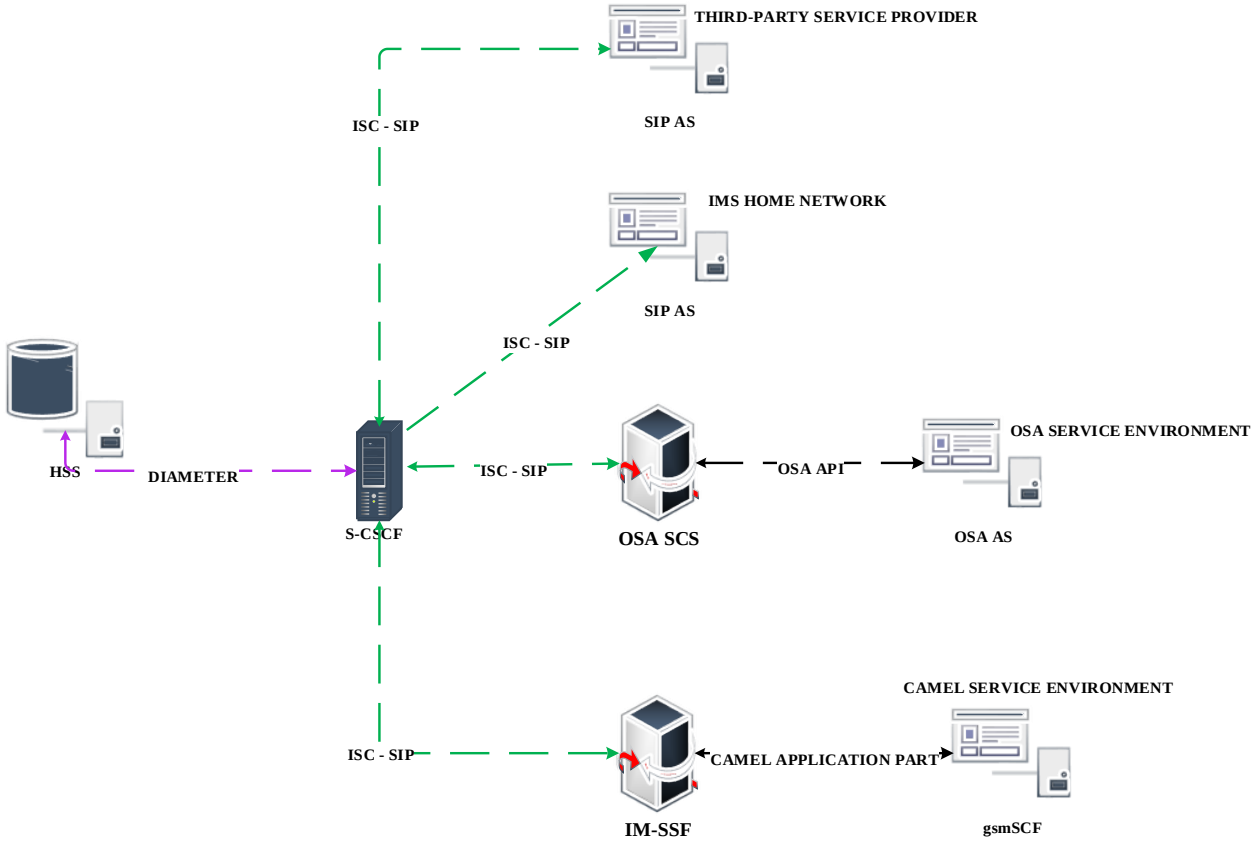

Figure 3-1 - The IMS service architecture. Adapted from Khlifi & Grégoire (2008).

Figure 3-1 also shows that SIP is the main signalling protocol through which service interaction takes place, thus the IMS Server Control (ISC) interface is standardised to define the routing of SIP messages to ASs (Reichl et al., 2006). The ISC provides an easy way to manage service logic and integrate various types of services that are supported by the different types of ASs.

3.2.1. The SIP Application Server

The SIP AS enhances the ability to provide a modular architecture where different service providers can deploy one or more services onto a common IP core. It is a native IMS AS that provides signalling capabilities for handling the execution of a service where different components are invoked in response to SIP signalling requests (Khlifi & Grégoire, 2008). These invocations occur based on the interactions between key elements within the IMS architecture and may result in the AS assuming various server roles: redirect; proxy; originating user agent; terminating user agent or back-to-back user agent. The establishment of a multimedia call session for instance, involves the use of the S-CSCF and the MRF during service execution where the S-CSCF applies filter criteria to decide which service will handle the call, and the MRF controls the media capabilities required for it.

The SIP AS can also reside in a third-party network, usually with service level agreements with the network operator, and plays an important role in adding new services to the IMS network (Khandelwal, 2007). There are various SIP-based techniques that can be used to achieve this goal, such as the SIP Servlet API or the SIP Common Gateway Interface (CGI) (Khlifi & Grégoire, 2008). The SIP Servlet API is a Java API that defines a converged servlet model that permits the mixing of SIP and HTTP applications, while SIP CGI defines a CGI model for SIP that is somewhat inherited from HTTP CGI. Although these technologies are not expressly covered, it is important to note them here as they are part of a more general discussion on extensions to SIP-based networks, particularly the kinds of extensions that help create a service platform that is based on a convergent architectural model that leverages HTTP to create novel, integrated applications.

In addition, the ISC interface also facilitates this service extension through its ability to adapt to interactions with ASs outside of the operator domain, hence breaking the concept of the walled garden (Higa, 2008). There is a common view that telco networks operate in this manner where access to their devices, platforms and equipment is restricted to outside entities. However, the ability for the SIP AS and the other ASs as shown in subsequent sections to facilitate external access to the IMS contradicts this common view. Bertin, Crespi & L'Hostis (2011) are also of a similar opinion where they refer to the argument of the telco network as a closed network to be a myth.

3.2.2. The OSA Service Capability Server

The OSA SCS is an AS that acts as a gateway between the IMS and ASs based on the OSA framework, whereby connectivity between OSA servers and the OSA SCS is provided through the OSA API (3GPP, 2008a). The OSA API is jointly defined by the Parlay Group, 3GPP and ETSI where the Parlay Group specifies a set of interfaces that are independent from the underlying network technology, since the specifics of the underlying network are the responsibility of both the 3GPP and ETSI. The OSA SCS translates instructions sent through SIP messages into a format understood by the OSA API to provide access to OSA-based services whose logic resides in an OSA AS (Moerdijk & Klostermann, 2003). The OSA SCS is typically located in the home network, whereas the OSA AS can be located externally in a third-party service provider network, or on the open Internet. Therefore, the OSA SCS and the SIP AS perform similar functions in terms of communicating with the S-CSCF via the ISC to invoke services. In the same way as the SIP AS, the OSA SCS can also interact with the MRF to define media interactions and how the OSA platform is to incorporate the IMS service capabilities with their service enablers.

3.2.3. The IMS – Service Switching Function

The IM-SSF is an AS that acts as a gateway between the IMS and services that implement the Customised Applications for Mobile networks using Enhanced Logic (CAMEL) standard, which are used in Global System for Mobile communication (GSM) networks. Camarillo & Garcia-Martin (2007) show that CAMEL-based services operate over legacy Intelligent Network (IN) infrastructure, and as such, implement non-IMS protocols for session control involving IN service capabilities. For example, while the SIP AS uses the Diameter protocol to interface with the HSS, the IM-SSF uses the Mobile Application Part (MAP) when doing the same. As a result, the interaction between the IM-SSF and IMS ensures that GSM-based functional entities can thus provide services to users, whereby the GSM Service Control Function (gsmSCF) handles service logic (Ghadialy, 2004).

3.3. Insights from Application Server Functionality

From the discussion on the IMS ASs, the following requirements can be made in terms of conceptualising how the IMS service architecture fits in with the overall aim to integrate with external networks such as WebRTC networks.

1. Gateway functions as bridges to external networks

There is a strong historical context for integrations with IMS involving external systems. The OSA SCS and IM-SSF give evidence of the exposure of service capabilities to third-party networks. These functions act as ASs on one side and as gateways on the other by describing interfaces and protocols to external networks that may not adhere to IMS standards but are translated to IMS-based protocols, as exemplified by using SIP when interacting with the S-CSCF and the OSA API or CAMEL when interacting with the OSA framework and the GSM network respectively.

2. Reuse of IMS functionality

Since the implementation of Common IMS, the 3GPP has developed a systematic way of enabling third-party access to IMS where any new requirements that may emerge from the integration are handled by working groups and liaisons that are appointed to oversee extensions to IMS. For instance, when integrating with the OMA, a release package was created that described their definition, requirements and architecture for services employing PoCC, messaging, conferencing, presence and availability and many more (Open Mobile Alliance, 2005). As such, third-parties can have their services integrated in a coordinated way which demonstrates an important business case for the IMS.

3. Use of internal and external protocols

The internal use of SIP within the home network ensures consistent development of the IMS service environment while external interfaces such as the OSA API are suited for IMS integration with other networks. Given that integrations with other networks are not uncommon, IMS also adapts to suit these interactions. For example, the implementation of SIP through the ISC interface ensures that the different ASs are able to interact directly with the S-CSCF, thus enabling third-party access to specific functions while also enabling adaptability to external interfaces.

3.4. Standardised Interfaces between Participating Entities

The telco service layer expanded further to include the GSMA as the standardisation body in charge of implementing web service-based APIs. Haas & Brown (2004) describe a web service as a web application that uses HTTP and eXtensible Markup Language (XML) as underlying technologies for its use, access and description in order to support interoperable interactions between endpoints over a network. Initially, telcos implemented CAMEL-based IN services whose implementation was typically

restricted to their network, and as a result, demand for “more innovative programming paradigms for service platforms” (Magedanz, Blum & Dutkowski, 2007) was prevalent within the industry, hence the adoption of service interfaces that provided a high-level abstraction from the underlying network. In fact, this ability to abstract away from the platform led to the proliferation of API implementations that rested upon the notion of service providers reusing existing investments in components such as ASs within their architectures to create extensible service delivery platforms.

According to Khlifi & Grégoire (2008), the OSA API enables the network operator to support APIs based on programming practices that result in protocol/platform-independent access to the components employed when executing services. Thus, the evolution of the Parlay OSA API to Parlay X led to the rapid development of applications using web services. Furthermore, interfacing mechanisms such as the Java APIs for Integrated Networks (JAIN) were developed as an alternative means through which service management and execution could be supported. JAIN provides an efficient application execution environment that supports the creation of integrated service mashups through the JAIN Service Logic Execution Environment (SLEE) (Tsietsi et al., 2015).

Still more, Internet-wide demand for more efficient API standards for third-party service interaction led to the emergence of Representational State Transfer (REST)ful web services. RESTful-based web services are modern and lightweight due to their efficient use of HTTP concepts in their architectural style where interactions between clients and servers leverage HTTP methods to exchange data. Thus, the telco’s adoption of the OMA Next Generation Service Interfaces (NGSI) and GSMA OneAPI as dominant API implementations within the field of RESTful web service-based APIs points to the significance of the pervasive influence that the web is beginning to have on telecommunication networks. The NGSI framework details APIs for data configuration and management, call control and configuration, multimedia list handling, context management, service registration and discovery and identity control, whereas OneAPI enables global operators to create applications that are written for mobile networks interoperable across multiple networks (Tsietsi et al., 2015). Figure 3-2 shows a framework of the overall telco network and the different APIs.

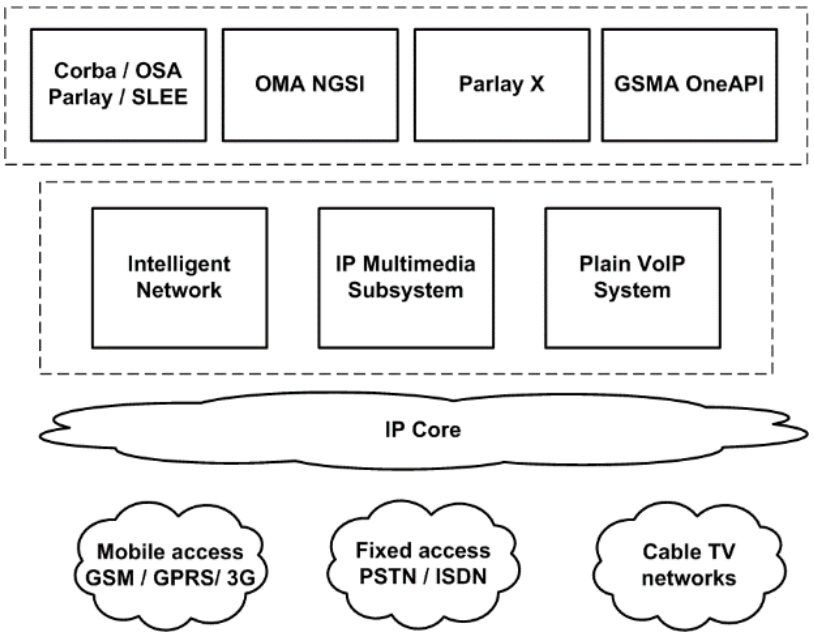

Figure 3-2 - Telco API overview. Source: Tsietsi et al. (2015).

3.5. Conclusion

The 3GPP collaborates with other standardisation bodies to define how to open telecommunication networks to third-parties. This integration is realised through standardised interfaces that enable ASs and the core IMS entities to interact with each other, thus showing how these interfaces are crucial in positioning the IMS as an integration platform. As such, providing common capabilities accessible to different kinds of user terminals, even those served by different platforms such as the OMA, is made possible through API implementations in the form of OSA API (nee Parlay X), OMA NGSI and OneAPI which are notable exemplars. The aim is that a service interaction of this sort could enable “secure, standards-based and billable access to services” (Tsietsi et al., 2015) where the network operator natively supports these services over their IMS architectures without having to rely on third-party infrastructure. To this end, the next chapter will analyse seminal work from the 3GPP in the form of a key technical report which investigates, in great detail, the possible architectural models that can help realise the integration between IMS and WebRTC, as well as the key design questions that are central to such conversations.

4. Chapter 4 – The Integration of IMS with WebRTC: A Review

4.1. Overview

This chapter aims to discuss the architecture and the mechanisms that are necessary to support the integration between WebRTC and the IMS service layer. The previous chapter facilitated a discussion on the functional roles of standard IMS service layer elements and demonstrated what is necessary to support the integration by contextualising the extent to which existing elements are re-imagined or have their roles and functions re-defined. In recognition of the need to chart a clear path toward the goal, the 3GPP through an existing working group, investigated the architectural implications of providing WebRTC access to IMS. The result was the drafting of TR 23.701 which is a technical report that emerged out of this investigation. The report proposes several integration models, each one describing a qualitatively different way of combining WebRTC and IMS (3GPP, 2013). It represents exploratory work that proposes underlying protocols and techniques to be used to facilitate the integration and asks some critical questions when assessing potential telco interest. The discussion on architectural models culminates in the introduction and analysis of the 3GPP reference architecture for WebRTC integration. Subsequent chapters will detail how the model that is proposed in this thesis borrows from specific architectural alternatives including the reference architecture but is constrained by the objectives of the investigation.

4.2. Requirements for a Basic Integration Architecture

The previous chapter highlighted three important features of the IMS service layer: the use of gateway functions as bridges to external networks; the re-use of IMS functionality, and the use of internal and external protocols. Thus, if follows that the integration model must fulfil these fundamental requirements to determine the readiness of the IMS to leverage WebRTC.

Requirement 1 - the use of gateway functions as bridges to external networks

Chapter 3 introduced the AS as the entity responsible for hosting and executing services, and where necessary, becoming a gateway function to connect external domains to the IMS. Similarly, the integration with WebRTC requires the definition of functional nodes that have their roles re-imagined to handle WebRTC-specific extensions in order to eliminate or minimise modifications to standard IMS elements. This effort to minimise the extent of the modifications imposed is an important consideration given that modifications have the potential to adversely affect other (possibly unrelated) IMS processes, or otherwise complicate the integration of such features into existing equipment or software, making them less practical. Furthermore, Shores et al. (2014) emphasise in their description of the methods to adopt when extending IMS to HTML5 environments that the importance of developing a system that does not require extensive or fundamental modifications to the browser model (where HTML5 is one of the main standards upon which WebRTC is based) is paramount. As such, the AS as the foundational entity for the development of an additional or re-imagined mediation function results in an architecture that is simple, effective and does not require expertise to develop applications to use the system.

Requirement 2 - the re-use of IMS functionality

Benali et al. (2004) mention that the ability to deploy new technologies (such as WebRTC) over operator networks requires that these technologies are realised "with reduced capital and operational expenditures in order to maintain sustainable growth of the whole industry and society" (Benali et al., 2004). Therefore, the main advantage of mediation functions such as the P-CSCF or an SBC is the ability

to evolve the network incrementally to integrate with WebRTC, thus resulting in re-usable architectures, components and frameworks. Moreover, there is great opportunity in retaining investments in quality developers who have extensive knowledge of the web and telco ecosystems that tend to be complex and cumbersome to gauge.

Standards Development Organisations are investigating each environment and their potential integration forms to tap into systems whose infrastructure is continuously being adapted to solve real problems through communication services. Even though the WebRTC and IMS integration use case is still ambiguous, under-specified and lacks a fully interoperable framework, there exists further potential for research and standardisation efforts to deliver appropriate models. For instance, Bertin et al. (2013) suggest either extending IMS to enable inter-working with WebRTC-based functions using gateways or creating a new telco control plane that is cloud-based and supports Infrastructure-as-a-Service (IaaS); Service-as-a-Service (SaaS); and Mobile Virtual Network Operators (MVNOs) and other ways of abstracting the network using WebRTC as the driving technology.

Requirement 3 - the use of internal and external protocols

The implementation of standardised interfaces ensures that entities can communicate using protocols and mechanisms that conform to pre-defined requirements and security considerations determined by standardisation bodies. The integration landscape therefore involves a wide array of these protocols that are translated and converted by mediation functions. Furthermore, inter-connection with the web domain results in the creation of an innovative space that will allow for even more flexible mechanisms that can be easily abstracted at various levels. As such, *Requirement 3* is split into *Parts a* and *b* to cover a discussion of the communication protocols supported and the utilisation of the WebRTC API respectively.

Part a - the implementation of internal IMS protocols

Rosenberg et al. (2011) explore the ability for browsers to support basic operator network protocols to enable interoperability at levels that go beyond the reliance on mediation servers. Both WebRTC and IMS employ protocols such as UDP, Transmission Control Protocol (TCP), ICE and RTP, although WebRTC employs the secure RTP profile, hence ensuring a common framework that requires little modification to the overall architecture as previously discussed, and reduces challenges experienced by gateways when performing media conversions. Furthermore, the WebSocket protocol as the main communication channel for WebRTC messages uses existing HTTP infrastructure, which IMS also supports. However, differences in the protocol suites are evident, for example, when WebRTC adopts a key exchange algorithm such as DTLS to secure media and data channels whereas the IMS mainly uses SDES.

Still more, the ability to support data exchange in the integrated scenario adds complexity. The WebRTC Data Channel is used to transport arbitrary data such as text messages, files and photos. Jesup et al. (2015b), also suggest using it to exchange control plane information between peers to enable signalling, conferencing, gaming and other use cases. Within IMS, the Binary Floor Control Protocol (BFCP), the Message Session Relay Protocol (MSRP) and T.140 (a real-time text presentation layer protocol) can perform similar functions to the Data Channel. BFCP is used to manage the way applications access a set of resources common to participants in a conference, where floor control determines whether users have shared or exclusive access. For instance, the protocol instils requirements that can enable a user to send media to a particular media session and not the other (Miniero et al., 2008). MSRP on the other hand is used within a SIP session, typically in the RCS context

and as with voice and video sessions, uses SDP to negotiate messaging capabilities between clients (O'Connell, 2007). Further, T.140 is a text format used in the context of a Global Text Telephony (GTT) environment where real-time text conversations take place either independently or in combination with other media. It can also be used in conjunction with IMS SIP to realise its functionality and therefore support interoperability with other networks (ITU-T, 1998). Thus, inter-working the Data Channel with IMS could foster applications that are yet to be explored or are still under investigation.

Part b - the use of external protocols

The ability for WebRTC to provide readily available solutions in the form of a standardised web API lowers the barrier of entry for developers wishing to experiment with browser-based RTC capabilities that were previously too complex and cumbersome to implement using plugins. With the browser as the main access platform for WebRTC, the network operator is thus able to provide services ubiquitously over devices that are not solely limited to the web. In fact, the use of the WebRTC API aligns with the adoption of the RESTful web service-based interfaces such as OSA API and OMA NGSI that enable an efficient external interface to the IMS network. On the same note, the demonstrations conducted by companies such as Google and Ericsson show the innovative ways in which the WebRTC API components are being used to develop applications based on voice, video and arbitrary data exchange.

4.3. Basic Integration Architecture

Dynamic client applications are created and run over lightweight web server functions whose functionality can be mimicked in the integrated scenario by IMS AS functions suitably adapted to support service provisioning and interoperability with WebRTC. For this purpose, a client is an application, running on a WebRTC-enabled browser, capable of accessing IMS services hosted by an AS and running over User Equipment (UE) (Muswera & Terzoli, 2010). A UE is any device with which a user can interact with a client (a hand-held telephone, laptop, personal computer etc.) and offer access to multiple access networks with the ability to roam. Figure 4-1 illustrates an integrated network where the question marks symbolise a collection of mediation functions that need to be put in place to enable the integration of WebRTC with the IMS service architecture.

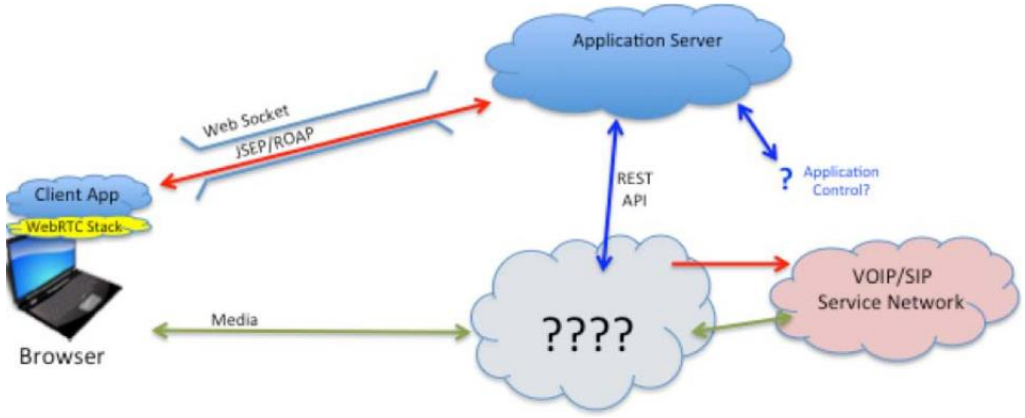

Figure 4-1 - Basic integration architecture showing. Source: Sansay (2013).

4.4. Solutions Analysis

The analysis that follows discusses and interrogates each solution as proposed by the 3GPP in technical report 23.701. The report includes certain assumptions that have been made about the architectural requirements for each solution and these are listed as follows: first, that SDP is used for negotiation

media parameters; second, media multiplexing is not supported and if used by WebRTC clients, the IMS network would remove the portion of the SDP offer that is associated with media multiplexing and third, that minimal modifications be made to the IMS network when enabling WebRTC access to IMS. Therefore, WebRTC-based media extensions are handled by inter-working functions. Lastly, the report also describes the use of functional entities handling NAT traversal, charging and billing policy control to enable QoS support. Each solution is analysed according to three key aspects: architecture, registration and session handling scenarios. While the discussion on architecture identifies the entities involved, the registration and session handling scenarios detail the interactions that occur between these identified entities. The use of this approach allows the discussion to highlight or emphasise the similarities and differences between each solution. In addition, missing requirements can be easily identified, leading to the possibility of synthesising a suitable integration architecture based on specific criteria.

4.4.1. Solution 1

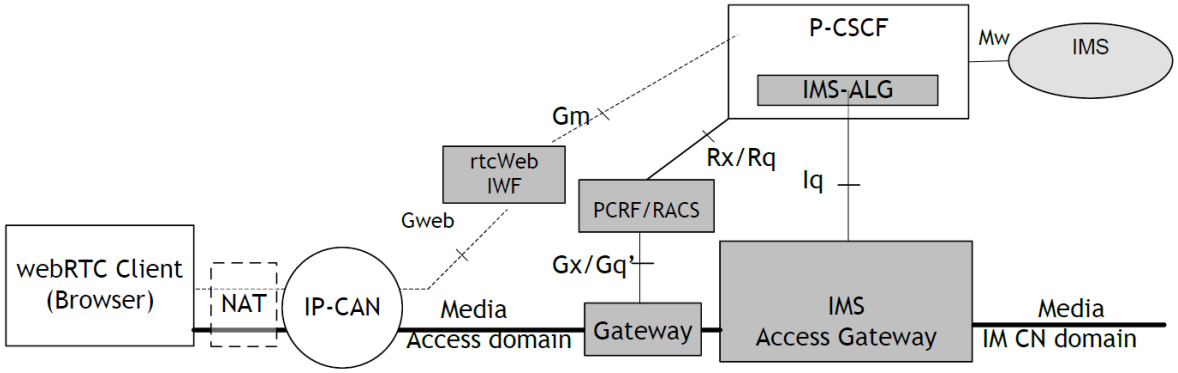

Figure 4-2 -Solution 1 architecture. Source: 3GPP (2013).

4.4.1.1. Architecture

Solution 1 depicts a generic integrated system showing an RTCWeb Inter-Working Function (IWF) mediating the control plane, with the IMS Access Gateway (AGW) mediating the media plane. The IWF in IMS is responsible for providing signalling interworking between the IMS network and a service provider that may be using a different signalling protocol to SIP (Brouquet, 2008). The *Gweb* reference point represents any of the signalling alternatives described in *Section 2.7.3* and is thereafter translated into a format that conforms to the *Gm* reference, symbolising SIP, which the P-CSCF propagates towards IMS. The IMS AGW is a functional entity that resides in the home IMS network and is responsible for reserving resources that are to be consumed during a media session, where either client can communicate behind a NAT or firewall (Camarillo & Garcia-Martin, 2007). During signalling and session negotiation, the P-CSCF requests a transport address from the IMS AGW which then reserves an address for the requested media flow and sends that to the P-CSCF for inclusion in the control path. Consequently, the IMS AGW routes the media packets appropriately towards clients participating in the session. The solution also provides support for IP version 4 (IPv4) and IP version 6 (IPv6) translation performed by the IMS Application Level Gateway (ALG) co-located with the P-CSCF.

The solution depicts the IMS AGW working in conjunction with the *gateway* shown next to it in Figure 4-2. From the diagram, the author gleans that this function applies IP forwarding policies to media to provide differentiated WebRTC services with these policies informed by the Policy Control and Charging Rules Function (PCRF), a “functional element that encompasses policy control decision and

flow based charging control functionalities" (3GPP, 2008b). The implementation of *Gx/Gq* and *Rx/Rq* reference points has the ability to further proliferate the creation of state-of-the-art integrated business models (Pascual, 2014). This thesis does not consider QoS constraints, along with NAT and firewall traversal using ICE connectivity because the Policy Control and Charging (PCC) framework is extensive and requires independent specification and investigation beyond the research scope as per *Section 1.5*.

4.4.1.2. Registration Scenarios

4.4.1.2.1. Registering a WIC using IMS Digest-Based Authentication

The scenario, as shown in Figure 4-3, begins with the WebRTC IMS Client (WIC) registering with IMS via the RTCWeb IWF over a WebSocket connection. The IWF then converts this connection to UDP, Transport Layer Security (TLS) or TCP when relaying signalling messages to the P-CSCF. In sending the "Register" message, the WIC maintains an identity binding by including its IMS Public User Identity (IMPU) as the username within the message but the solution does not discuss possible strategies to allocate the IMPU and associated credentials to the client. As such, opportunities exist to employ a web server either managed by the network operator or a third-party in a trust relationship with the network operator in this process. Friese et al. (2010) suggest various ways of integrating the Internet and web identity strategies to support identity provision where, for instance, the browser can store the user's identity information via browser cookies or adopt the HTML5 Web Storage API in the client application to allow for storage that is more persistent following an initial subscription to IMS. These browser-based mechanisms represent alternatives to the Subscriber Identity Module (SIM)-based identity management schemes that are pervasive in IMS. Having propagated the "Register" message to the IMS core, the scenario follows the basic IMS registration flow (Khandelwal, 2007).

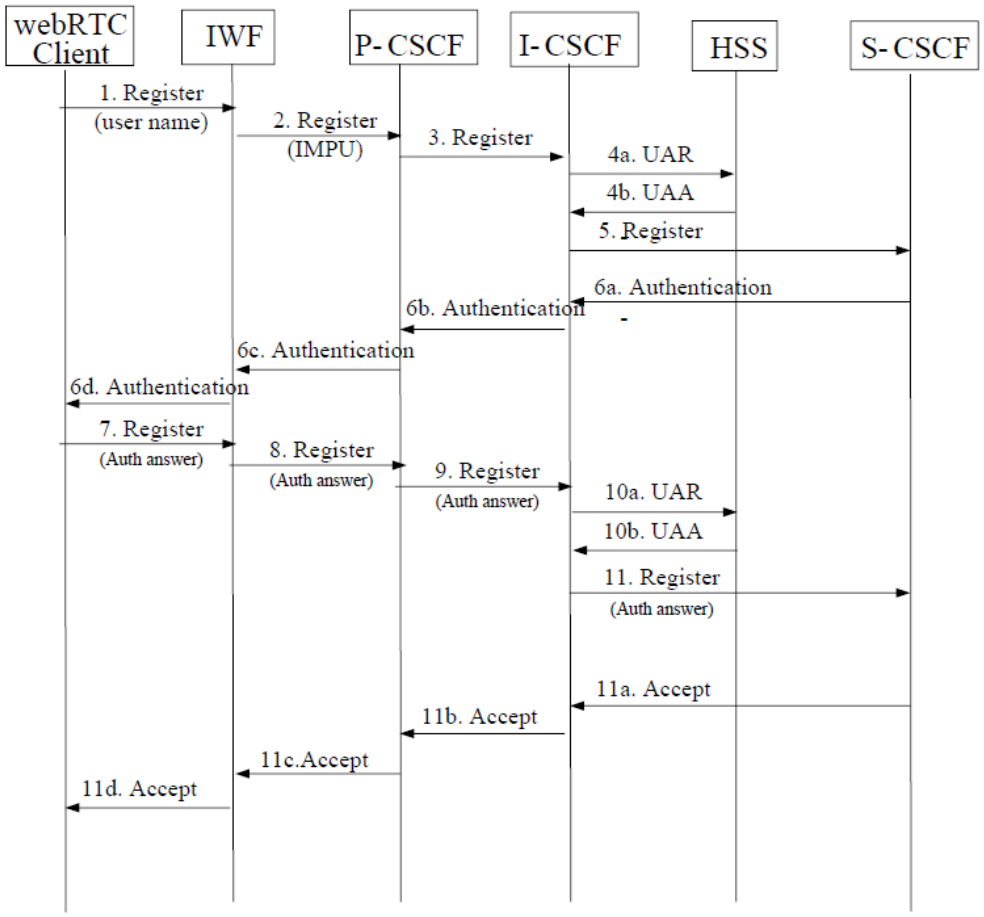

Figure 4-3 -Registering a WIC using IMS digest-based authentication. Source: 3GPP (2013).

4.4.1.2.2. Alternative Registration where the RTCWeb IWF acts as an IMS user

The call flow depicted in Figure 4-4 is an alternative to the one shown in Figure 4-3, where in this case, the RTCWeb IWF acts as an IMS user by performing third-party registration on behalf of the client. The process is identical to IMS registration and results in the IWF receiving an IMPU. When a client registers with the IWF, it includes its username and appropriate credentials within the "Register" message. Once authentication is successful, the IWF creates a binding between the username and the specific IMPU allocated to the user. This procedure is evident in the case of the IWF acting as an IP Private Branch Exchange (IP PBX) unit. An IP PBX is an entity that typically resides at the network edge and is used to switch phone calls between users residing in the same domain while also relaying control messages and requests between different domains (Prasad & Kumar, 2011). It is predominantly used by enterprises in order to handle client download, identity authentication and location management functions thus applying appropriate business level policies via standardised interfaces. Solution 6 which is presented later in this chapter suggests the use of WebRTC-based IP PBX emulation functions.

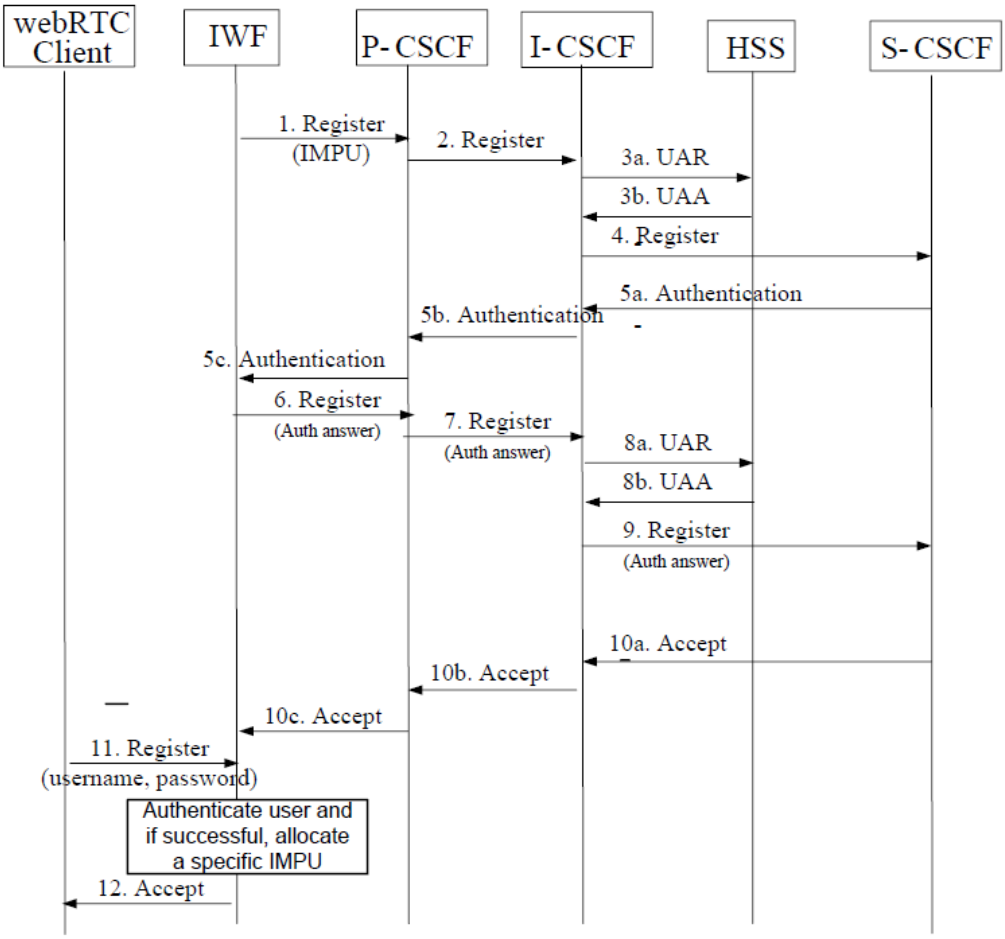

Figure 4-4 - Alternative registration process. Source: 3GPP (2013).

4.4.1.3. Session Handling Scenario

In the session handling scenario as depicted in Figure 4-5, a basic multimedia call occurs between a WIC and a standard IMS client with either client having the ability to originate and terminate the call. The WIC sends a "Setup Session" request to the IWF with the address of the target user included within the request. The IWF then sends an "Invite" message to the P-CSCF which follows regular IMS session setup procedures. Once confirmation is received that the session can begin, media flow occurs via the IMS AGW where the necessary media inter-working is performed.

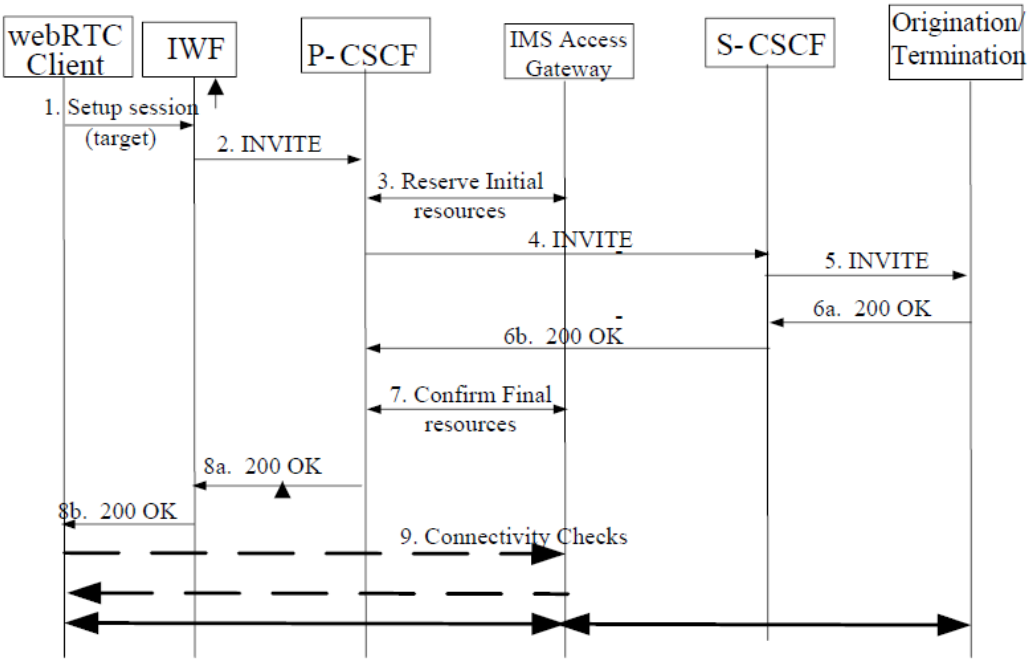

Figure 4-5 - Session handling between a WIC and an IMS UE. Source: 3GPP (2013).

4.4.2. Solution 2

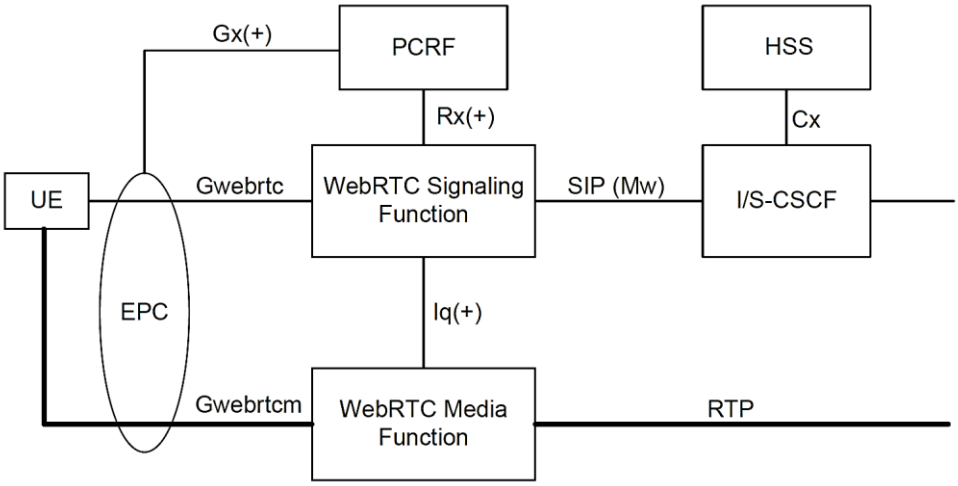

Figure 4-6 - Solution 2 architecture. Source: 3GPP (2013).

4.4.2.1. Architecture

Solution 2 decomposes the generic IWF introduced in Solution 1 into a WebRTC Signalling Function (WSF) and the WebRTC Media Function (WMF). Although not explicitly illustrated in the diagram nor mentioned in the report, the IMS AGW is an important entity that is required to handle the IMS-side media inter-working, therefore, it is safe to assume the inclusion of such an entity, or a similar one, within the IMS network.

The WSF comprises a signalling component, an SDP mediator and a Media Function Controller (MFC). However, additional components are supported, such as an ICE Agent to handle ICE connectivity. The signalling component is responsible for converting WebRTC-side signalling with the session negotiation capabilities being handled by the SDP Mediator. The SDP Mediator is also necessary to translate any extensions to SDP messages that the client may need to add to support WebRTC for example, the use of media multiplexing via RTP/RTCP into one port. WebRTC supports multiplexing whereas IMS does not, therefore the integration architecture would have to adapt its SDP interactions accordingly. The MFC coordinates with the WMF during session management to control media resources as well as to apply appropriate congestion control schemes when reserving such resources - the RTP mediator within the WMF is responsible for resource reservation and performs media conversion. Other components within the WMF include a transcoder, responsible for media codecs conversions in addition to an ICE Agent.

The ability to design components to support WebRTC extensions results in innovative architectures that could be co-located with different IMS entities, for instance, the WSF can be co-located with the P-CSCF while the WMF with the IMS AGW. Such an arrangement would also require enhancing the interfaces between these components in order to support efficient communication mechanisms that are more relevant to the WebRTC domain and to enable the WebRTC-based IWFs to process them more effectively. As such, there is a potential to standardise the *Gwebrtc* and the *Gwebrtcm* (representing media) interfaces particularly given that the WebRTC-side signalling scenario has been intentionally left ambiguous and is left to the will of the implementer.

4.4.2.2. Registration Scenarios

4.4.2.2.1. Registration using SIP over WebSockets for IMS authentication

The WIC initiates the registration process by sending a “Register” request via SIP over WebSockets to the WSF along with a username and associated credentials to validate and authenticate the user. The user obtains this identity information through means outside the scope of the solution. On behalf of the WIC, the WSF then includes the user's IMPU in the “Register” request sent to IMS for basic authentication. On the other hand, the WSF can simply indicate within the request that the user is part of the trusted domain and therefore does not require further authentication. The registration procedure ends as normal with the I-CSCF forwarding a success response to the WSF. The assumption is that this scenario is supported in addition to the ones depicted in Solution 1 where basic IMS registration is to be supported by all communications provided over the IMS network. Figure 4-7 shows the call flow for the registration procedure.

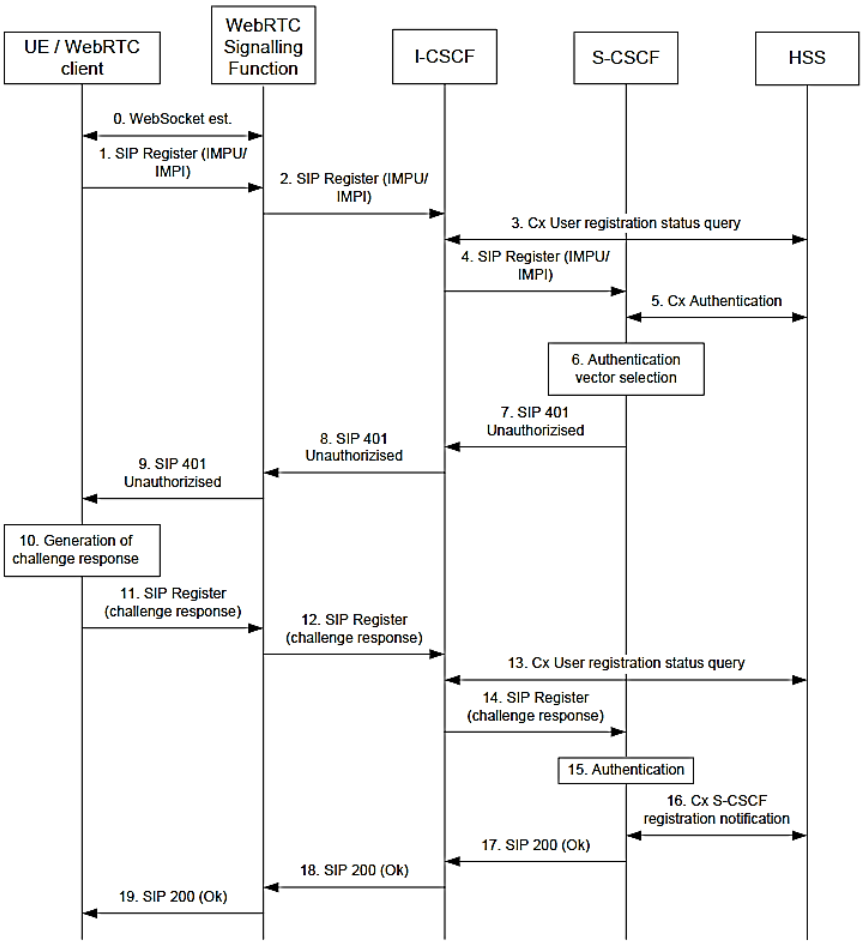

Figure 4-7 – Registration using SIP over WebSockets. Source: 3GPP (2013).

4.4.2.2.2. Registration using Web Authentication

This registration scenario enables a user to authenticate with IMS using web identities through web authentication schemes not specified within the solution. The WSF is responsible for receiving a user’s web credentials and issuing them with an access token once authentication is successful, and further performs the necessary mapping of the user’s web identity to their IMPU in order to continue with

IMS registration. Even though the access token issuance is not specified within the procedure, the authentication nodes employed during registration namely the WSF are trusted by IMS entities.

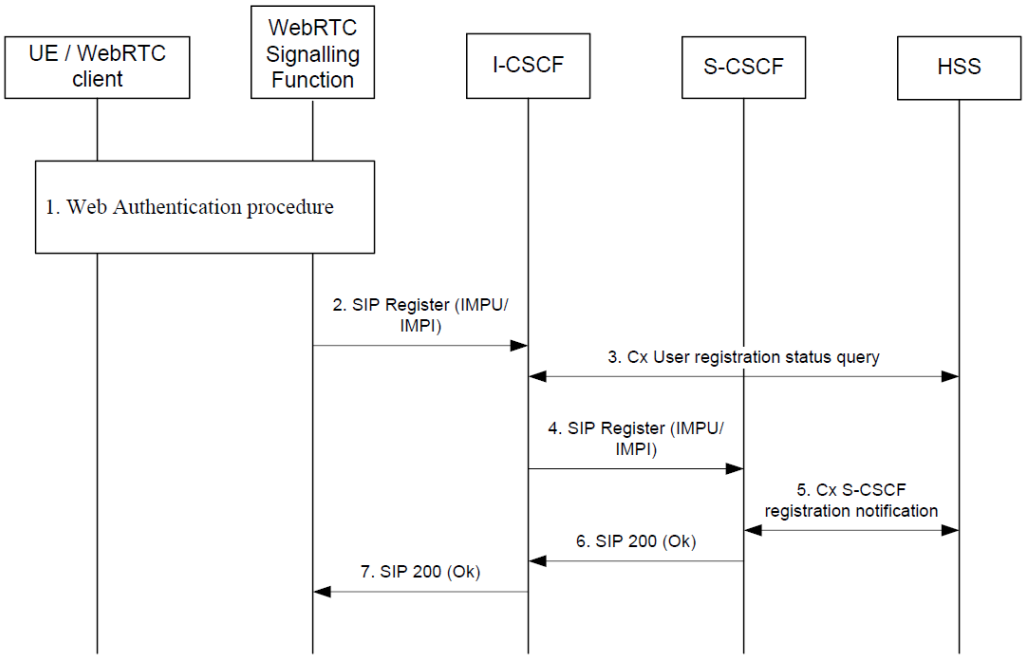

Figure 4-8 - Registration using web authentication. Source: 3GPP (2013).

4.4.2.3. Session Handling Scenario

The session handling scenario follows a similar model to Solution 1 where, instead of the RTCWeb IWF, the clients forwards the "Setup Session" request to the WSF that then propagates relevant "Invite" messages to the IMS core network and back to the WebRTC domain. Media is similarly handled as outlined in Solution 1 however, the WMF is included in the communication path, in addition to the IMS AGW.

4.4.3.Solution 3

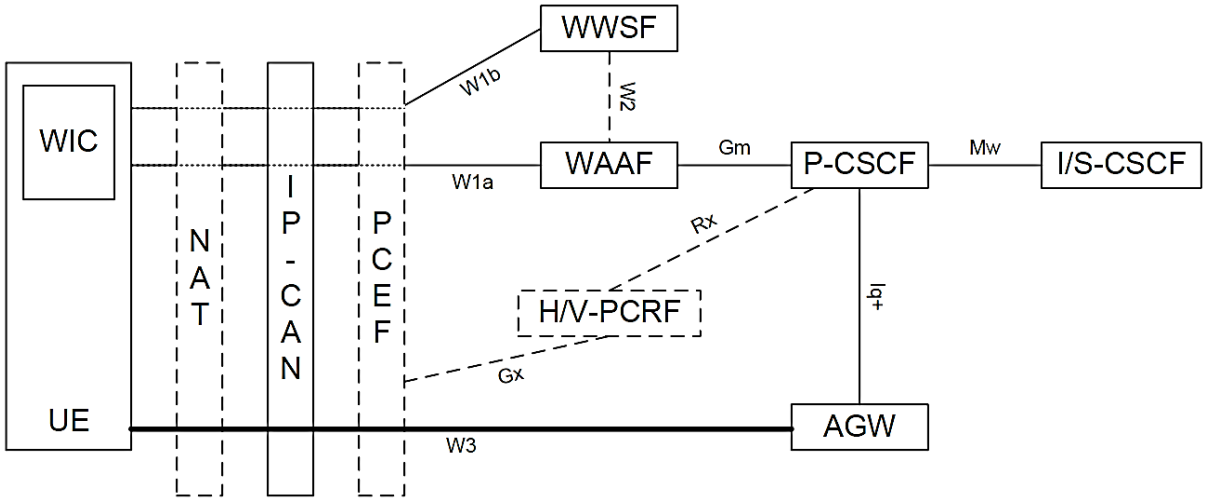

Figure 4-9 - Solution 3 architecture. Source: 3GPP (2013).

4.4.3.1. Architecture

Solution 3 introduces the WebRTC Access Aggregator Function (WAAF) and the WebRTC Web Server Function (WWSF). The WAAF is an IWF that performs advanced features, in addition to signalling translation. It performs identity management by communicating with the WWSF to allocate IMS identities to the user and acting as a SIP Registrar when authenticating those IMS identities – a function that occurs in consultation with the P-CSCF as outlined in the discussion on registration which follows. The WAAF also aggregates the signalling messages that are sent by multiple clients to the P-CSCF in an efficient manner. Furthermore, when the WAAF is located in a third-party network and provided the WWSF is also in that same network, it can optionally provide communication services to the user to enhance their experience. Within the solution, the WAAF and WWSF perform the majority of their functions together, hence, the *W2* reference point links them.

The WWSF on the other hand simply hosts the IMS services that the user subscribes to and accesses these services via web pages, therefore, it is the initial point of contact a user has with IMS. The WWSF can either be located in the home or third-party network and can perform advanced features in combination with the WAAF such as identity management as previously mentioned. As a result, it applies web authentication procedures to register a WIC with the network by maintaining a consistent binding between their web and IMS identities, thereby putting the necessary security measures in place.

4.4.3.2. Registration Scenario

The introduction of the WAAF and WWSF emphasises the importance that the solution places on utilising web-based schemes to manage user identities and provide enhanced services, hence their combined functionality. As a result, the description of the registration scenario is extensive and expresses procedures in terms of how they differ in the authentication method: digest-based IMS authentication, web authentication and wild-card IMPU; type of IMPU being registered and ownership (typified by location) of the WAAF and the WWSF. Figure 4-10 to Figure 4-13 illustrate the call flows for user registration.

4.4.3.2.1. Registration using IMS Digest

This scenario depicts a digest-based registration process when a WIC registers an individual IMPU via the WAAF located in the home network. The WAAF is restricted to the home network to prevent man-

in-the-middle attacks that digest-based authentication is easily susceptible to, whereas the WWSF can either be located in the home or third-party network. A secure connection is established using HTTPS between the WIC and the WWSF in order to authenticate the user's IMS credentials by interacting with the WAAF via the *W2* interface which forwards their identity information to the appropriate IMS entities. The WIC then opens a secure WebSocket connection to the WAAF using Cross-Origin Resource Sharing (CORS) procedures to ensure that the WIC is served by a trusted WWSF that has been authorised to serve it. Following which, a "Register" request can then be sent to the P-CSCF via the secure connection.

Fette & Melnikov (2011) state that CORS is a mechanism used by the WebSocket protocol when connecting clients and servers, whereby a server can reject a script that comes from an unknown (essentially untrusted) origin and as a result, ensures that requests are received from entities trusted by the network to perform the authentication. According to Sansay (2013), it is a mechanism that needs to be properly handled in order to avoid the potentially large security risks of implementing WebRTC and IMS gateway functionality, particularly when interfacing with an external web server. It is the responsibility of the WAAF to translate the transport mechanism from WebSockets to the appropriate IMS protocol that can be understood by the P-CSCF and other core IMS entities. The process then follows standard IMS registration procedures that once are successful, allow the user to gain access to IMS communication services.

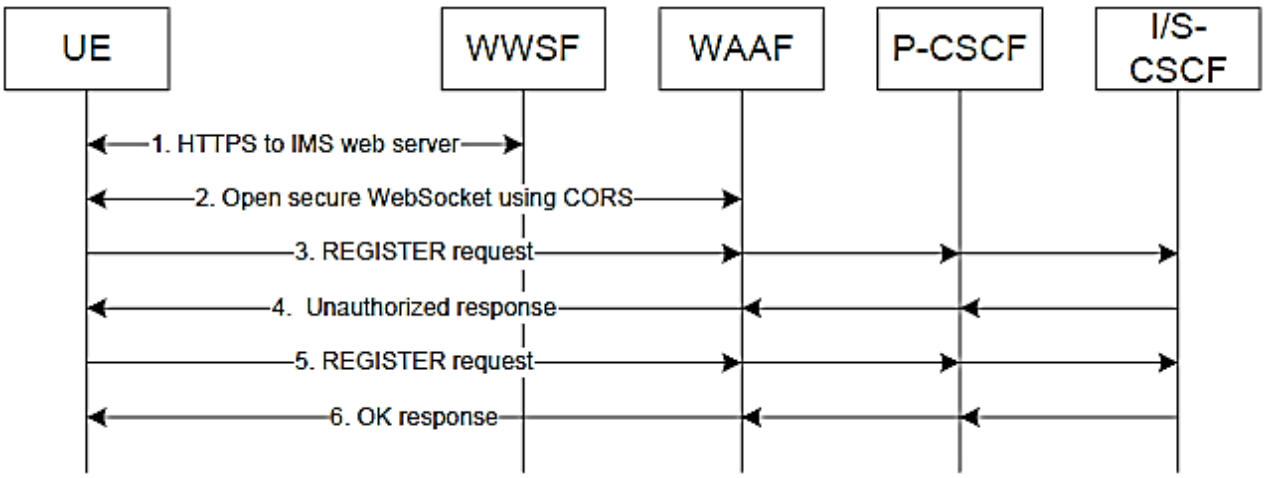

Figure 4-10 – Registration using IMS digest. Source: 3GPP (2013).

4.4.3.2.2. Registration using Web Authentication

This registration scenario describes the interaction between the WWSF and the WAAF in the authentication of IMS users using their web identities. It is similar to the corresponding one described in Solution 2 in that it is the preceding step depicting how a user obtains a mapping of their IMS identity based on their web identity (a process which was out of scope for Solution 2 given that the architecture does not show interactions with a web server function).

The client initiates the registration process by establishing a secure connection to the WWSF and provides their user information when logging onto the service. A security token is then issued to the client containing the user's IMPU. As in the previous scenario, the WIC establishes a secure connection to the WAAF using CORS, after which an authentication-less registration process ensues where the

WAAF informs the P-CSCF that the WWSF has already authenticated the user during the issuance of the security token. Because the WWSF is a trusted authentication node, IMS successfully registers the user. The use of the WWSF as a trusted entity borrows from Jennings, Peterson & Watson (2002) who describe the ability of certain SIP servers to authenticate and assert user identities within a restricted domain.

The WAAF must strictly reside in the IMS network for it to completely trust the authentication-less registration request from the user, while on the contrary, the WWSF can be located either in the home or third-party network even though it is also involved in the authentication process. Moreover, the WAAF is a registrar server and is therefore the one that performs the crucial validation step checking the security token received from the client to make sure that the IMS identities being registered are from an authorised WWSF.

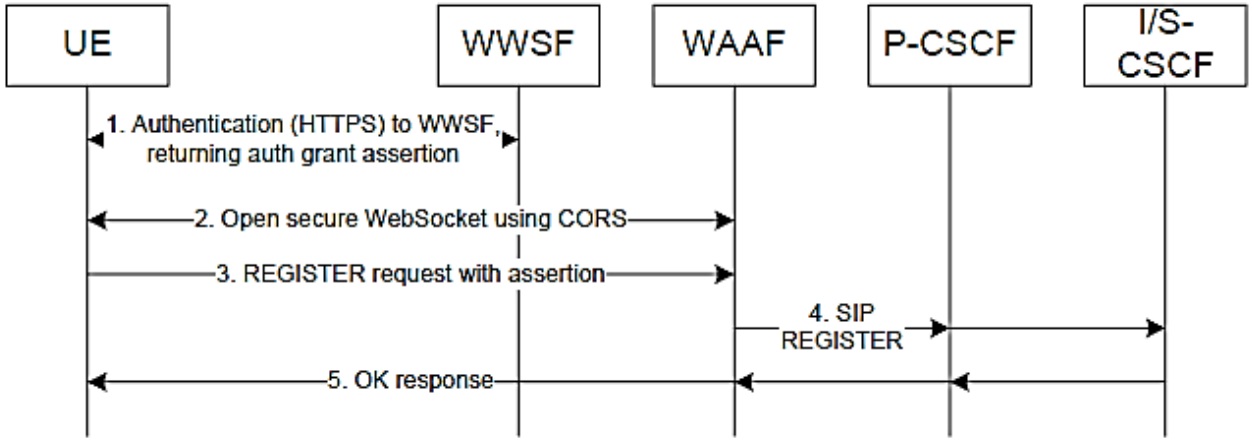

Figure 4-11 – Registration using web authentication. Source: 3GPP (2013).

4.4.3.2.3. WAAF Registration of Wildcard IMPU with IMS on behalf of WWSF

This scenario is an extension of Solution 1 where it shows the RTCWeb IWF registering an IMPU on behalf of the clients it serves. In this case, the WWSF is responsible for obtaining a range of IMPUs that it allocates to the pool of clients it serves. The effect of employing the WWSF and WAAF during this process results in differing registration modes that the IMS applies depending on the entity the WAAF interfaces with. For instance, when the WAAF interfaces with an Inter-connection Border Control Function (IBCF) or IP PBX, the IMS pre-registers the IMS identities in the wildcard IMPU range during user terminal configuration to hide the terminal configuration from the third-party network. Brouquet (2008) defines an IBCF as a function that may be adopted between two different enterprise domains, typically employing SIP, to enable communication in such a way that the networks hide their configuration to protect them from security vulnerabilities. It can also obfuscate SIP headers and other information about the network such as the number of S-CSCFs, their capacity and the capacity of the network. The IBCF may also integrate with an IWF to enable interoperation with other signalling protocols such as WebRTC-based ones. On the other hand, the WAAF interfaces with a P-CSCF when the IMS dynamically registers the identities. For this scenario, the location of the WAAF is not restricted to the network to enable the third-party domain to provide value-added communication services on top of the IMS service offering.

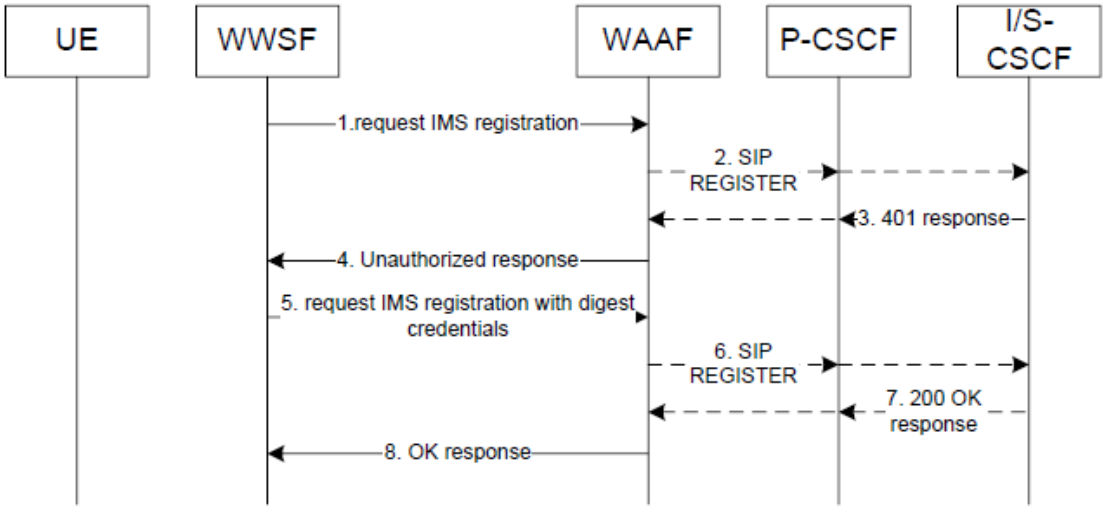

Figure 4-12 – WAAF registration of wildcard IMPU. Source: 3GPP (2013).

4.4.3.2.4. WIC Registration of Individual IMPU from Range

Once the range of identities have been registered according to the preceding scenario, individual WICs can then follow the web authentication procedure shown in Figure 4-13 below with a similar process to *Section 4.4.1.2.2*. The difference is that the WAAF is responsible for verifying the user's identity assertion and authorising their access to services. It is also able to verify the third-party registration on behalf of the user by either examining the configuration data attached to the user's identity or based on a prior arrangement with the WWSF to register the range of identities.

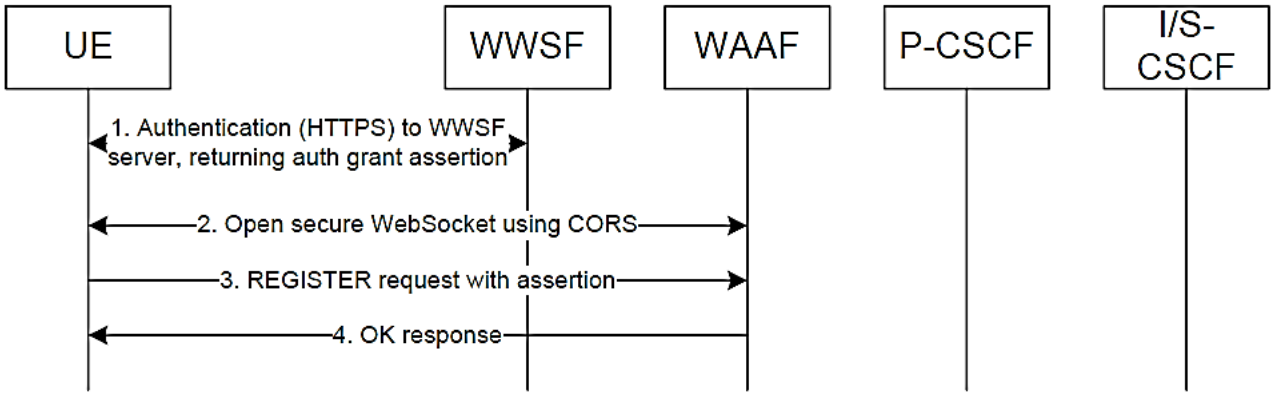

Figure 4-13 - WIC registration of individual IMPU from wildcard range. Source: 3GPP (2013).

4.4.3.3. Session Handling Scenario

The session handling scenario on the other hand, follows the standard IMS procedure where the WIC, WAAF, P-CSCF and other IMS entities are involved in the session establishment path.

4.4.4. Solution 4

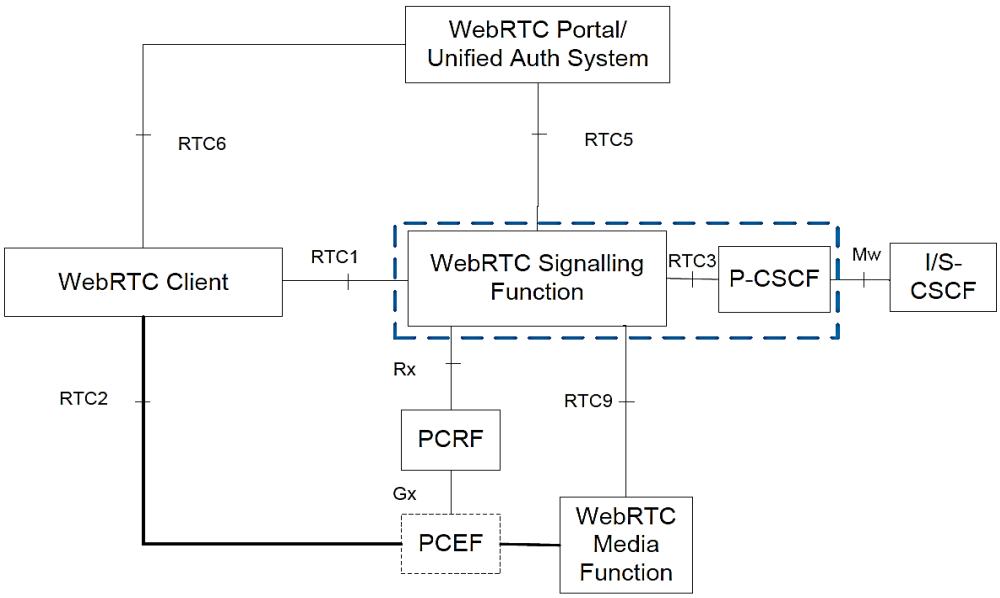

Figure 4-14 - Solution 4 architecture. Source: 3GPP (2013).

4.4.4.1. Architecture

As with Solution 3, Solution 4 places particular emphasis on the authorisation and authentication of users subscribed to IMS, hence the introduction of the WebRTC Portal/Unified Authorisation System which may be located either in the home or third-party network. Furthermore, previous solutions also present components, the WSF and WMF whose function is familiar. In this architecture, the difference is that the WebRTC Portal System functions similarly to the WWSF, while the combined WSF and P-CSCF functions similarly to the WAAF.

The purpose of the Portal System is to ensure that authorised users access IMS services by providing means to verify their IMS identities via a web-based application also hosted by the Portal System. Moreover, it also informs a client of the IP address of the WSF serving it. The dual functionality of the Portal System is reminiscent of the WWSF seen in Solution 3 in that they are both responsible for managing user identification by authenticating users and mapping their web identities to their IMPUs, while also hosting the JavaScript code containing application content.

The solution also suggests that the WSF may be co-located with the P-CSCF, a feature that is also possible with Solution 2. The importance of co-locating these functions results in a modularised architecture that reduces the added complexity of managing the interactions between these entities when performing the necessary inter-working. At the same time, the combination of these entities also leads to possibility of evolving IMS entities to support web-based features, which is a concept that comes with its own complexities. This implementation is in favour of adding complexity at the gateway level, as opposed to the client level, in order to reduce barriers to application development, thus attracting the innovate web developer. More importantly, the solution is a segue to the reference architecture described in TSGC (2015) and discussed in *Section 4.6* which is based on an augmented IMS network that is enhanced to implement intelligent WebRTC mechanisms.

4.4.4.2. Registration and Session Handling Scenarios

The registration and session handling scenarios for this solution are similar to the ones seen in previous solutions where support for IMS and web authentication schemes are described, particularly those for Solution 3, albeit with differences in the nomenclature of the functional entities.

4.4.5. Solution 5

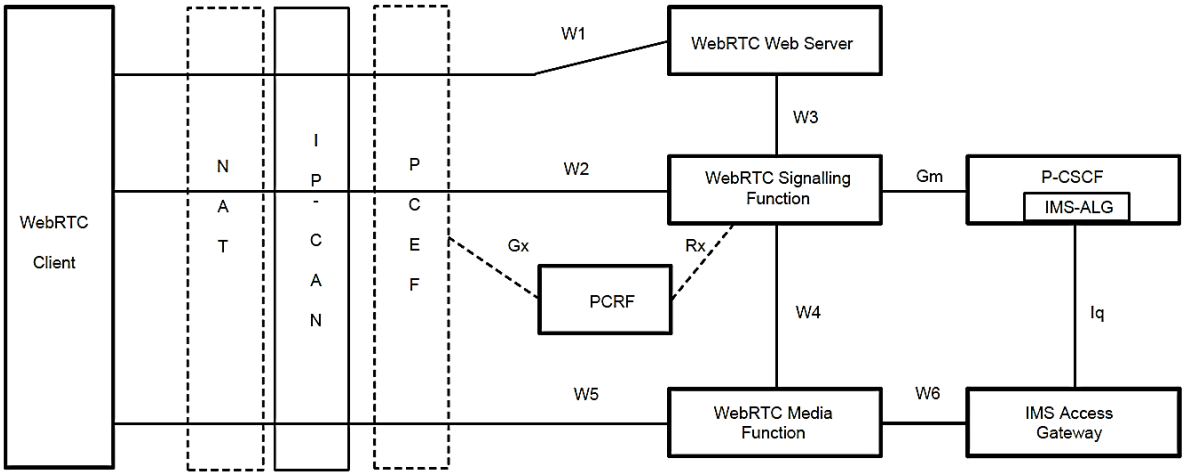

Figure 4-15 - Solution 5 architecture. Source: 3GPP (2013).

4.4.5.1. Architecture

The architectural arrangement for this solution is presented in Figure 4-15, and is evidently an amalgamation of some of the components seen in previous solutions. For instance, the WWSF has already been introduced in Solution 3 and the WebRTC mediation functions for signalling and media were seen in solutions 2 and 4. As such it is evident that the functions in this solution serve similar purposes to those already described. This architecture is simply more straightforward in its arrangement and use of terminology. As a result, it is conceivable that the architecture is a clear expression of integration with the WebRTC domain and at a superficial level, implies a more efficient realisation by the operator when providing web-telco mashups.

4.4.5.2. Registration and Session Handling Scenarios

As with Solution 4, the registration and session handling scenarios for this solution are similar to those seen in previous solutions where support for IMS and web authentication schemes are described, particularly those for Solution 3, with differences in the nomenclature of the functional entities. However, the solution explicitly mentions the use of operator-provided web identities and associated credentials that can be mapped to IMS entities as previously described. Figure 4-16 shows the registration call flow for this novel use case.

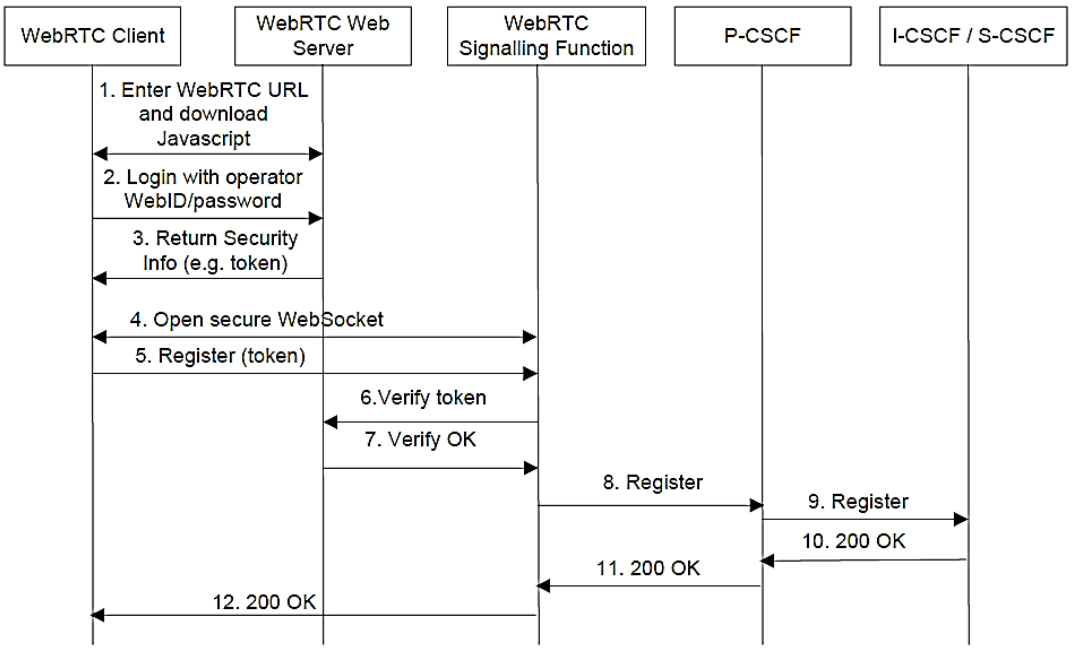

Figure 4-16 - Registration using operator-provided web identity. Source: 3GPP (2013).

4.4.6. Solution 6

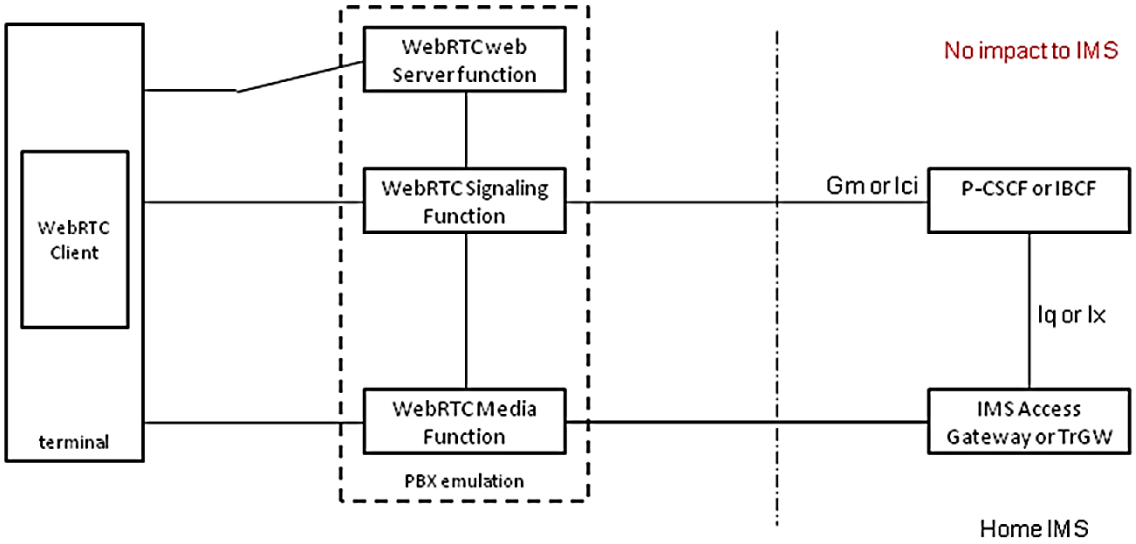

Figure 4-17- Solution 6 architecture. Source: 3GPP (2013).

4.4.6.1. Architecture

Solution 6 locates WebRTC mediation and server functions within an IP PBX emulation node that resides between two enterprise domains. The node is responsible for all interactions with the client, and therefore provides the necessary interfaces required to handle client application downloads from the WWSF; identity authentication, session establishment and location management by the WSF and media handling by the WMF. These functions are similar those seen in previous solutions and translate the WebRTC-based interfaces to IMS with messages propagated towards either a P-CSCF or an IBCF.

4.4.6.2. Registration and Session Handling Scenarios

The registration and session handling scenarios follow a similar pattern to the one described in Solution 1 where the IWF acts as an IMS subscriber and can therefore register on behalf of the client. The client would then be authenticated by the IWF and have its web identity mapped to its IMPU. Similarly, the WSF in this scenario registers with IMS and independently registers the client. Furthermore, the scenarios described for Solution 3 are also applicable to Solution 6 where, instead of the WAAF, the WWSF and WSF emulation functions are able to jointly authenticate and authorise the WIC to access IMS. Again, this functionality elects the IWFs as trusted authentication nodes in their capacity as registrar servers. This separation of concerns is necessary to abstract the network from the user, especially when interoperating with a different domain whose security profile may be unknown, and thus prove to be a concern for the network. Moreover, the functions are used to offer telephony services such as call holding, transfer and others that according to Prasad & Kumar (2011), can be conformed to standard IMS business trunking interfaces and procedures, where trunking refers to the ability to adopt business policies that inform the way different domains make connections between subscribers (3GPP, 2013).

4.4.7. Solution 7

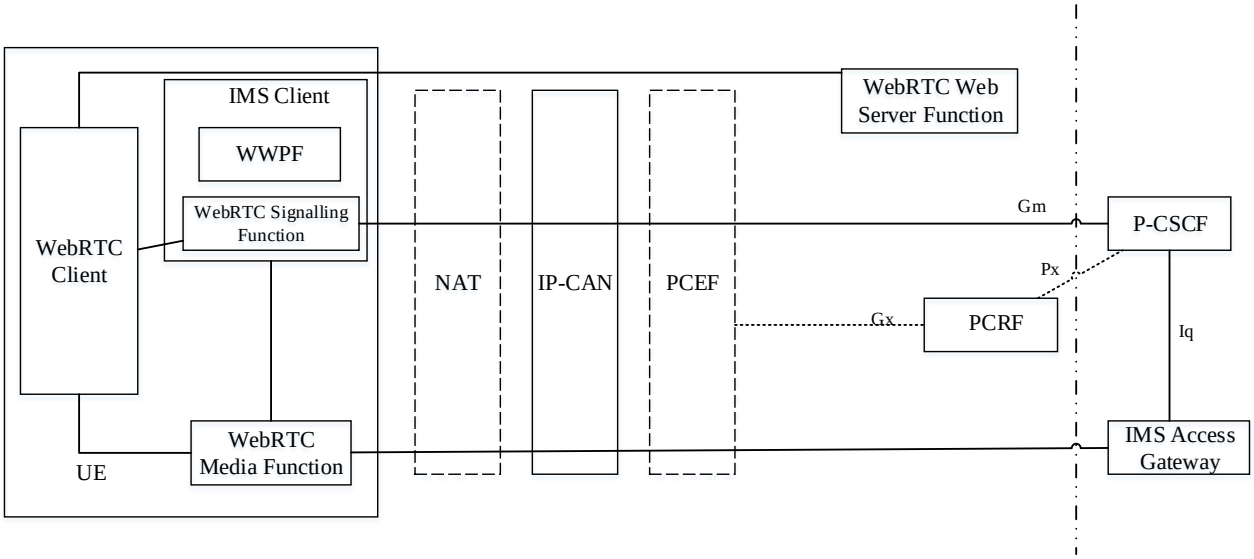

Figure 4-18 - Solution 7 architecture. Source: 3GPP (2013).

4.4.7.1. Architecture

The last solution presents an architectural arrangement that differs considerably from the ones previously seen through its co-location of a WebRTC client, IMS client, WebRTC signalling and media functions and an entity called the WebRTC Web Proxy Function (WWPF) into a single UE. The UE then interacts with a web server function and IMS core entities. As such, the functional entities, with the exception of the WWPF, operate as expected based on previous solutions. The main purpose of the WWPF is to provide an interface between clients in order to enable the browser to access the user's IMS credentials directly from a Universal Integrated Circuit Card (UICC) application provided by the SIM without user intervention. Consequently, this solution is applicable to the specific use case where the UE is a standard IMS user terminal that follows classical IMS registration procedures.

This design alternative strongly positions the telco as an IdP that has the capability of exporting the user's identity to the web. The solution does not provide support for web authentication procedures, however, in the case of a telco-operated architecture, it can be argued that the telco has the ability to provide the means to map a users' web and IMS identities, thus supporting the registration scenario depicted in Solution 3. On the other hand, the telco could also provide web identities and consequently support the registration scenario illustrated in Solution 5.

Even though the solution does not describe alternative ways to access user information, Solution 1 suggests storing it using web-based mechanisms such as the HTML5 Web Storage API that informs the possibility of implementing API support in the WWPF due to its pre-existing involvement in the acquisition of said user credentials. As a result, non-UICC based clients could still benefit, and in addition, operators would continue to have tighter control over their architectures especially when managing WebRTC functions, particularly the web server functions whose ability to support web identities proves most advantageous. Furthermore, supporting flexible web integrated clients further proliferates Minerva & Bell (2010)'s mandate of the ability of the operator to create open environments that "enable adaptive, overlay and self-organising technologies" (Minerva & Bell, 2010). The authors go on to suggest the possibility of implementing virtual networks on a global scale in the form of MVNOs as a business strategy for the operator and use the success of Apple in this regard as evidence.

4.4.7.2. Registration Scenario

The scenario begins when a user accesses application content on the web browser via a WebRTC client. As the WebRTC client has no direct interface with the UICC, the IMS client is thus responsible for accessing the user credentials because it already supports mechanisms to enable this access. IMS registration then follows the usual pattern. Figure 4-19 provides an illustration of client interactions with IMS and a web server managed by an operator.

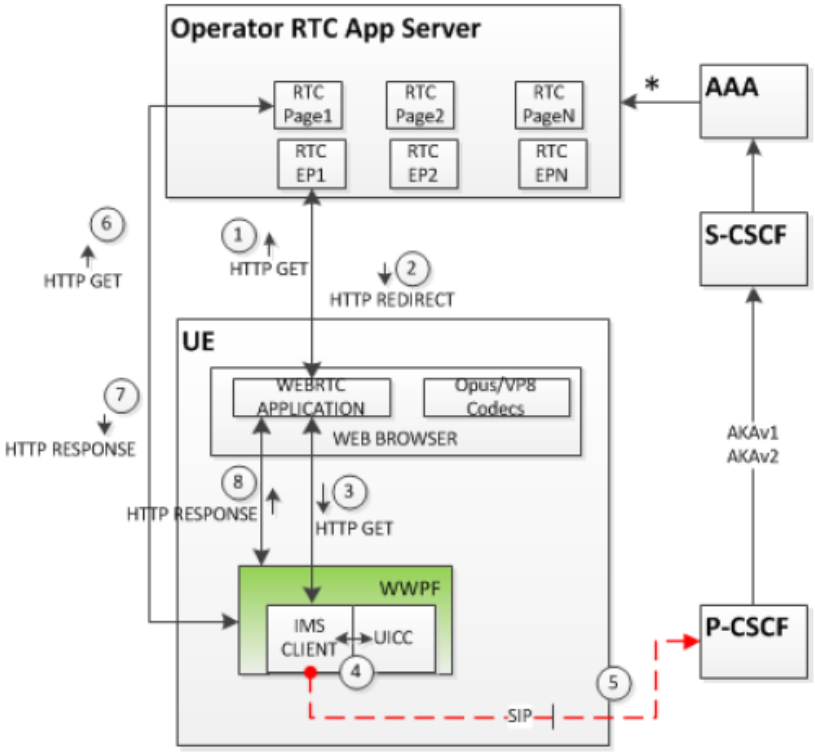

Figure 4-19- WebRTC authentication using IMS credentials. Source: 3GPP (2013).

4.4.7.3. Session Handling Scenario

The session call flow, depicted in Figure 4-20, shows detailed interactions between the components employed during transcoding, ICE connection management and general session and media handling. In the diagram, SIP messaging is handled by the Signalling Inter-working Function (SIF), which performs similar functions to the WSF, while transcoding and protocol conversion are handled by the RTC Media Inter-working Function labelled (RMF).

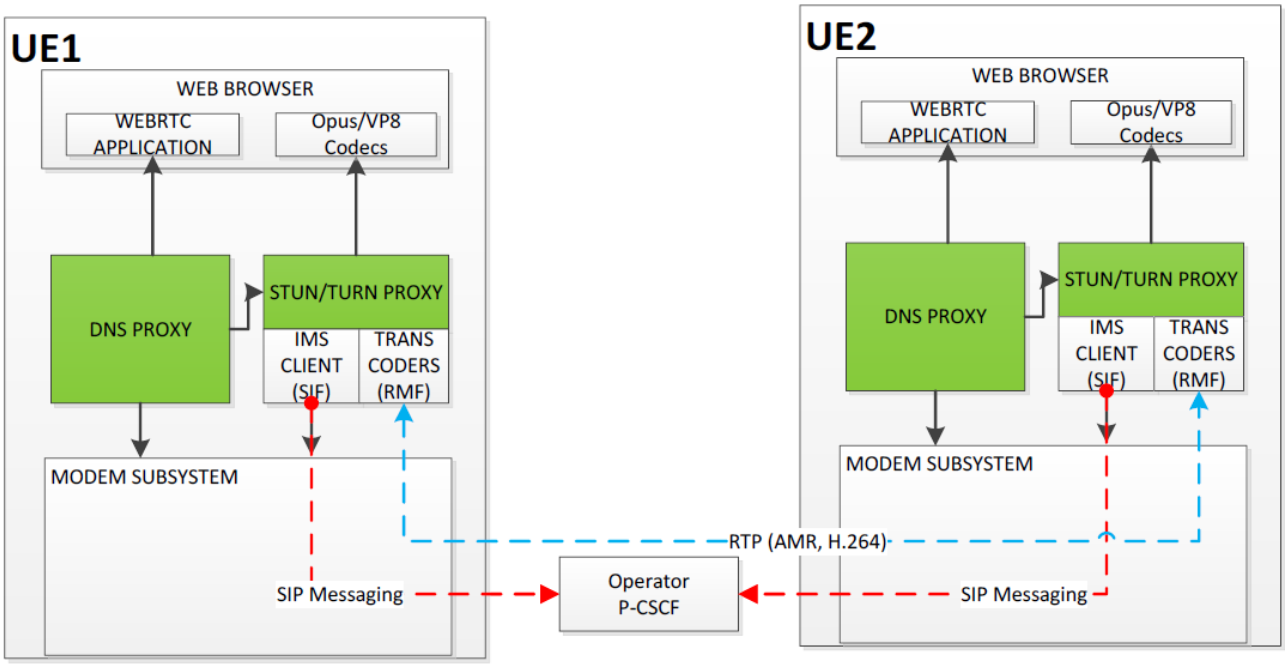

Figure 4-20 - Session handling on operator controlled WebRTC. Source: 3GPP (2013).

4.5. Insights from Solutions Analyses

Having analysed the solution architectures presented in the previous section, this thesis proposes two additional requirements for performing WebRTC and IMS integration to add onto the three existing requirements presented in Section 3.3.

Requirement 4 – the ability to create integrated clients and UEs

The different techniques supported by the integrated architecture ensure support for different combinations of clients in different scenarios where the operator or third-party is trusted to provide the service. Examples of support include the ability to integrate with existing operator-controlled service platforms and the ability to extend current UEs. These techniques are conceived to extend the IMS ecosystem and Raivio & Luukkainen (2011) believe that supporting strategies to "open up" the network results in increased innovation for the types of services and the resultant business models which attempt to find a balance between the closed walled garden ecosystem that is typical for the network operator and their open systems. This requirement is split into *Parts a* and *b* and expresses insights that have also been developed from a literature study that was conducted in conjunction with the report analysis.

Part a – the ability to integrate with existing telco service platforms

Numerous opportunities exist to integrate a WebRTC-IMS ecosystem with existing technologies already offered by telcos such as RCS, in order to cover a wide range of service capabilities depending on the gateway design (infrastructure) deployed. The use of WebRTC with RCS as the main example is attractive given that RCS was initiated as a technology that telcos could use to enable collaboration with the web. In fact, both technologies perform similar functions of creating an application-focused environment where contextualised (immersive) communication services are provided (Romain, 2013).

Part b – the ability to extend current UEs to the web domain

Johnston, Yoakum & Singh (2013) suggest the ability to upgrade existing VoIP/SIP/IMS UEs to accommodate WebRTC. The notion of upgrading existing phones opens a way for network operators to become device manufacturers outside of enterprises where SIP phones are typically adopted and has the potential to change the behaviour of how users interact with these phones, from performing basic voice/video calls to enabling immersive communication experiences that are also found on the web. This thesis advocates that such a UE should require the implementation of web technologies that adopt WebRTC as an RTC engine. The feasibility of such a device remains an open issue however, it also creates a space for the telco to become more involved in the original equipment manufacturer (OEM) value chain, in addition to the provision of integrated (device independent) web applications and services. According to Olanoff (2015), Google's acquisition of Jibe Mobile could be seen as a motivating factor behind the interoperability between WebRTC and RCS; moreover, Mavenir, a software-based networking solutions company, is an example of an industry effort already involved in the development of such a solution (Richardson, 2014). The strategic move of these companies aims to disrupt the market by providing services on the Android OS platform based on RCS and even though Google would be promoting RCS, WebRTC can be the enabler to create enriched web experiences.

Requirement 5 – the potential to natively support web-based techniques

The standards and techniques employed around web identity management and media handling have the potential to lead to evolutions centred on WebRTC where IMS entities can natively support these features. The preliminary work conducted in the report provides a basis for the ability to incrementally

grow and develop such an architecture and describes the issues around supporting these features in Parts *a*, *b* and *c*.

Part a – the use of operator-based web identities

Solution 5 suggests the use of Operator WebIDs authenticated using existing web authentication procedures that the network maps to a user's IMPU/IMPI. The use of web-based identities is a strategy that can be employed by the operator to open up their network and result in Single-Sign On (SSO) systems that use SSO protocols such as OAuth 2.0 and OpenID Connect (Beltran et al., 2014). Furthermore, employing such a strategy leverages the strong position that the operator has to authenticate users for services, hence the adoption of phone numbers in user verification for applications such as WhatsApp, Facebook, Yahoo Mail and other OTTs which, according to Beltran et al. (2014) are international and thus collectively managed by operators. The formulation of Operator APIs developed by network operators thus becomes the next logical step; Mulligan (2009) mentions that Open APIs have been created which can be used in conjunction with an API such as WebRTC to access video conferencing capabilities, presence for user online status (notifications as well) and text-to-speech technology for accessibility during instant messaging. Raivio & Luukkainen (2011) further suggest the creation of ecosystems based on network operator APIs that would provide device-independent alternatives to device-based ecosystems such as Apple's App Store, Google's Android Market (Play Store) and Nokia's Windows Store.

The API exposure also enables the network operator to access identity information (SIM-based or otherwise) that they can use to improve user experiences through big data analytics. Solution 7 is an example of an architectural model over which an API, imagined as the "Operator SIM Authentication API", can be implemented to authenticate WebRTC users over the web with the aid of the WWPF. This process would require interfaces to IWFs to organise contextualised information about the user (retrieved from HSS and S-CSCF in their capacity as registrars and signalling function) and their multimedia sessions (retrieved from the media function).

Part b – the use of web-based identity protocols

The interface between a WWSF and a WSF is crucial due to the novelty it brings to managing web identities where the WWSF can also integrate value-added services. This interface is illustrated as *W2* in Solution 3, *RTC5* in Solution 4, *W3* in Solution 5 and is implicitly assumed in Solutions 6 and 7. Shekh-Yusef & Pascual (2014) suggest the adaptation of SIP to implement an authorisation framework such as OAuth 2.0. The combined adoption of these protocols can thus be one way for the WWSF and the variety of signalling functions to serve the user with access tokens during authentication, as depicted in Solutions 2, 3, 4, 5 and 6. Furthermore, when mapping a user's web identity to their IMPU, the relevant function (WSSF, WAAF, WebRTC Portal System, etc.) can also implement technology such as the Lightweight Directory Access Protocol (LDAP) as means to perform lookups for the identity of a user, an organisation (in the case of inter-connecting enterprise domains), a file or other resources in a network (Howes, Smith & Good, 2003).

An alternative to the SIP OAuth 2.0 framework could involve the use of JSON Web Tokens provided by an operator-managed web server function (Lynch, 2011). The main benefit of using a JSON-based token format is that it is generic and can be used as part of a standard authorisation protocol such as OAuth 2.0 or OpenID Connect. An authentication scheme such as this provides a flexible means through which a telco-operated IdP could have the ability to communicate with numerous web server implementations that exist on the web. The use of JSON for authentication also supports service

integration through the creation of an SSO system outside of the Generic Bootstrapping Architecture (GBA)-based framework that is responsible for user authentication and prevalent in IMS, thus extensively widening the reach of the telco (Muranyi & Kotuliak, 2013). Furthermore, the standardisation of the WWSF and WSF reference point leads to the evolution of RESTful web service-based API implementations (where OSA API, OMA NGSI and GSMA OneAPI are notable exemplars).

Part c – the ability to handle signalling alternatives

The ambiguous signalling landscape of WebRTC which seemingly elects data exchange formats such as JSON and XMPP/Jingle to carry session establishment messages introduces obvious differences with the IMS landscape. Solutions 2, 4, 5, 6 and 7 illustrate a relationship between the P-CSCF and the WSF, which is evidence of this potential to enable opportunities for expansion where techniques such as SIP over WebSockets and JSEP are key illustrations of the ability to contextualise some IMS protocols, SDP included, to the web domain.

Part d – the ability to handle WebRTC-based media

The interfaces between a client, a WMF and an IMS AGW as shown in all the solutions are similarly crucial to media handling not only for transcoding and protocol conversion, but for the development of suitable control structures that both WebRTC and IMS need to implement. For instance, with the P-CSCF and IMS AGW typically acting as the initial points of contact with IWFs for session handling and media, there is a need to implement small changes in their functionalities in order to support the new WebRTC access type. Furthermore, the S-CSCF, acting as the main entity used during service provision, might also require minor augmentation in its structure to handle the browser-based WebRTC services whose web server functions need to be recognised and trusted by IMS.

4.6. 3GPP Reference Architecture

This section documents the process that yielded the selection of a reference architecture for WebRTC and IMS integration. The reference architecture is documented in TR 23.701 and combines some of the solutions described previous sections, while also providing a forecast into the future of the IMS ecosystem in incorporating WebRTC. It was later added to IMS technical specification (TS) 23.228, in an effort to recognise the evolution of the telco industry towards enabling WebRTC access to IMS. However, the architecture in TS 23.228 differs from the initial one presented in TR 23.701, as such, this section will discuss the nature of the differences.

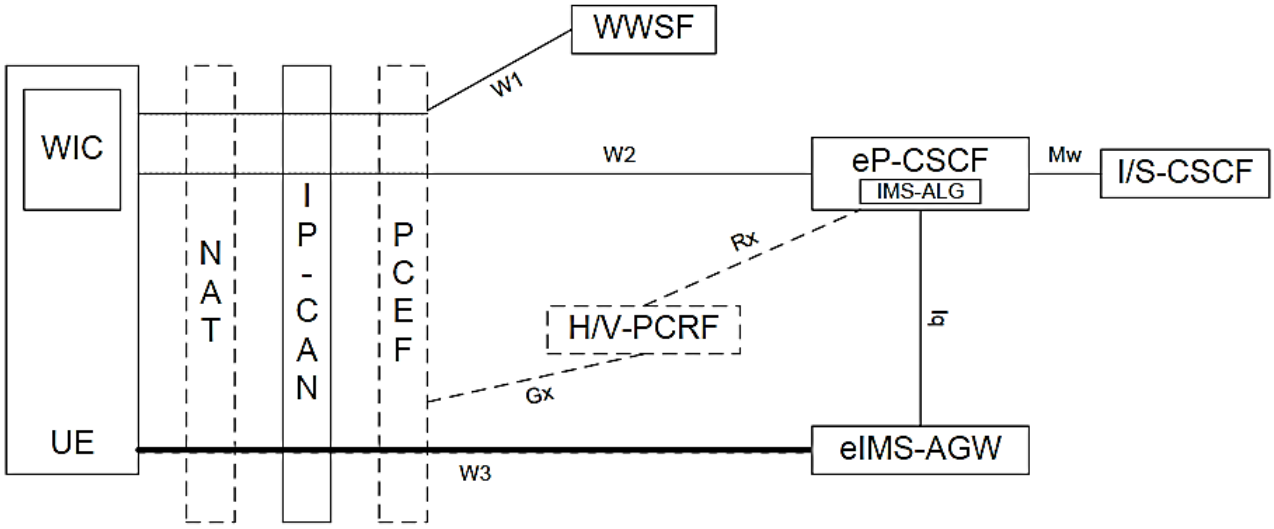

Figure 4-21 - 3GPP WebRTC and IMS reference architecture. Source: 3GPP (2013).

4.6.1. Architecture

The purpose of this architecture is to put together a model that shows interactions with IMS entities whose functions are extended to support interoperability with WebRTC, for instance the P-CSCF and IMS AGW termed the enhanced P-CSCF (eP-CSCF) and enhanced IMS AGW (eIMS AGW) respectively. The function of the WWSF has already been seen in Solutions 3, 5, 6 and 7 where a user interacts with a WWSF that can either be located within the operator or third-party network with the operator ultimately being responsible for its control and management. The WWSF is involved with user identities and is responsible for allocating and mapping the user's IMS and web identities. The eP-CSCF is responsible for authenticating users using IMS and web authentication means and authorising the verifications that the WWSF performs to ensure that identities are served by a WWSF trusted by the network. As such, the eP-CSCF is strictly located in the home network because of its role as a trusted authentication node. The limited functionality of the WWSF, compared to the aforementioned solutions, is due to the absence of a direct interface with the eP-CSCF.

The eIMS AGW on the other hand, is enhanced to support the media inter-connection typically performed by the WMF and is therefore required to support characteristics such as DTLS-SRTP, transcoding, DataChannel translation to protocols such as BFCP, MSRP or T.140 and other WebRTC functions. These media handling actions are carried out by the *W3* reference point in Figure 4-21 and represent an amalgamation of the *W3* in Solution 3, *RTC2* in Solution 4, W5 and W6 in Solution 5 and the similar unlabelled reference points in Solutions 6 and 7.

4.6.2. Registration Scenario

The registration scenarios for this solution are similar to the ones already seen, particularly in Solution 3 where IMS and web authentication schemes are used by the eP-CSCF which is the main authentication entity involved during signalling. The technical report depicts the following registration call flows shown in Figure 4-22 and Figure 4-23. Figure 4-22 shows WIC registration of IMPU using IMS registration while Figure 4-23 shows WIC registration of IMPU using web authentication procedures. The solution also provides the ability to provide a wild-card IMPU range to the WWSF from which individual clients can be assigned identities including the ability to support different registration modes. The architecture further describes the de-registration scenario, however, even though it is not explicitly described by other solutions, it is inherently supported and follows standard IMS procedures with messages traversing the relevant signalling functions.

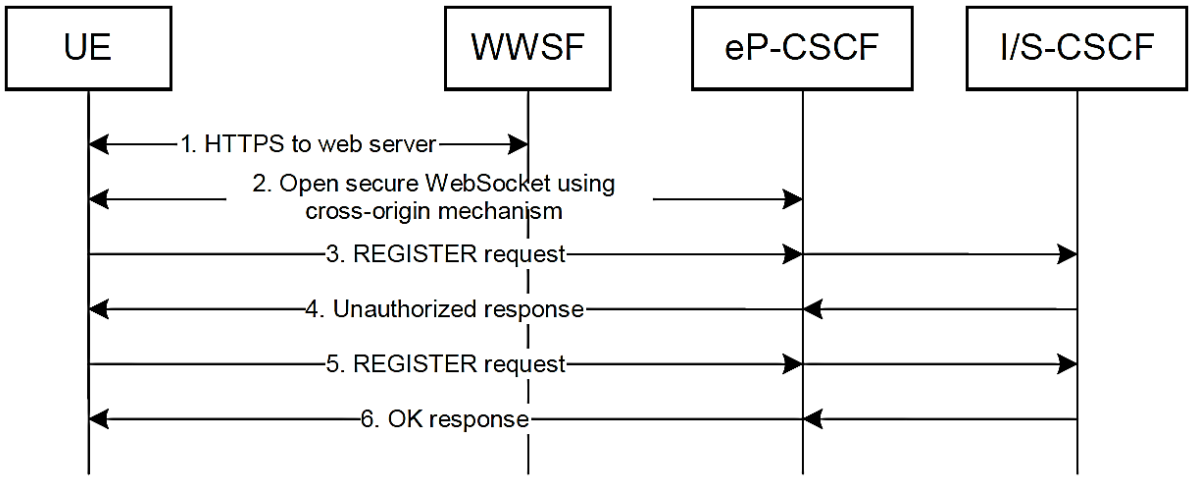

Figure 4-22 - WIC registration of IMPU using IMS registration. Source: 3GPP (2013).

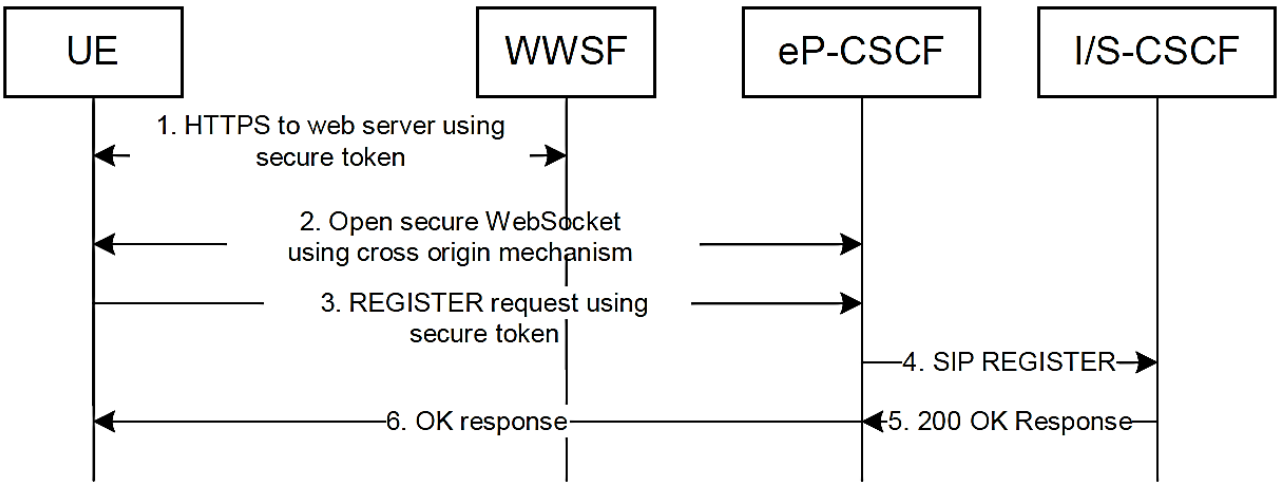

Figure 4-23 - WIC registration of IMPU using web authentication. Source: 3GPP (2013).

4.6.3. Session Handling Scenario

The eP-CSCF is responsible for routing session origination and termination flows between the WIC and other IMS entities according to standard IMS procedures. The architecture involves enhancing

reference points such as the *Iq*, *Mw* and others to incorporate WebRTC-based characteristics for efficient media flows.

4.7. Insights from the Enhanced Reference Architecture

Requirement 5 acts as a precursor to the enhanced architectural model where the eP-CSCF and eIMS AGW natively support web-based techniques such as web identities and media. Furthermore, the eP-CSCF has the ability to natively support WebRTC-based signalling protocols by inter-working them into native IMS SIP; functionality that was performed by the WSF in previous solutions. The shift in led to the eP-CSCF handling the combined roles of the WWSF and the WSF where a reference point was needed in order for these two entities to jointly support user authentication and signalling exchange as in Solutions 5 and 6. In other instances however, an additional component was introduced that independently handled the implementation of the reference point for example, Solution 3 introduced the WAAF and Solution 4 introduced the WebRTC Authentication / Portal Unified System that functions similarly to the WWSF. By the same token, Solution 4 also explicitly demonstrates a co-located P-CSCF and WSF operating together to handle the signalling inter-working and identity management which the author believes to have set a precedent to the enhanced P-CSCF.

In light of the above reflections, a future release of TS 23.228 re-imagined the reference architecture with the inclusion of an entity called the WebRTC Authorisation Function (WAF) (3GPP, 2015). The role of the WAF is similar to the combined functions of Solutions 3, 4, 5 and 6 where it is responsible for authenticating users using web procedures resulting in the issuance of access tokens to the WIC (via the WWSF) propagated towards IMS. The WAF can either authenticate users by itself or trust the WWSF to allocate users with their identities and similarly to the WWSF, can reside in the home or third-party domain. Figure 4-24 shows the updated 3GPP reference architecture showing the WAF whose registration and session handling scenarios are similar to the model shown in Figure 4-21.

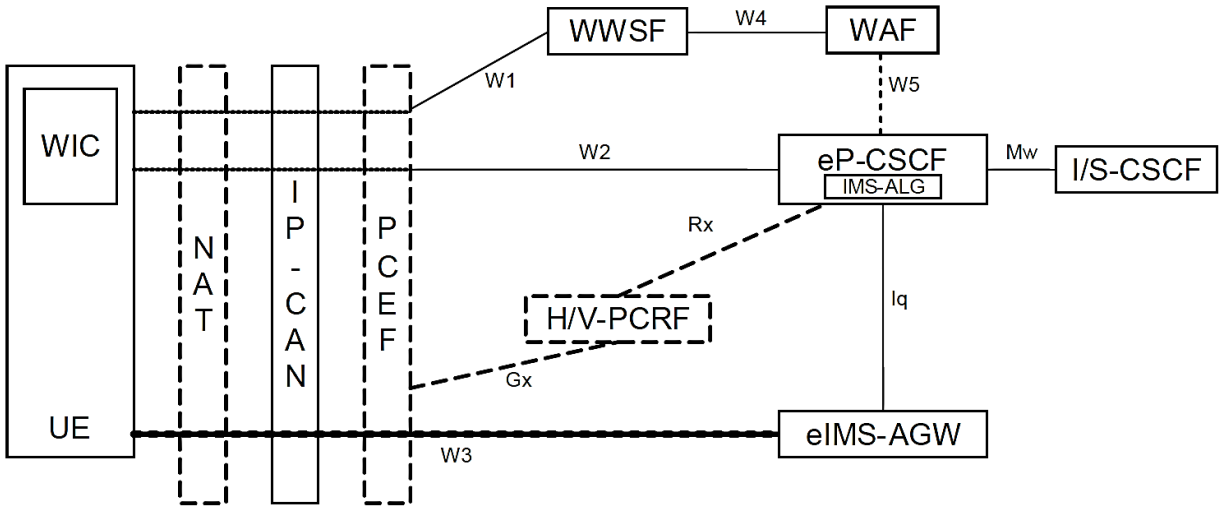

Figure 4-24 - Updated 3GPP reference architecture showing WAF. Source: 3GPP (2015).

4.8. Conclusion

The chapter brought about some emergent strategies defined by the 3GPP to assist telcos in expanding their IMS ecosystem to enable access to WebRTC. These strategies are evident in the seven qualitatively different solutions that are specified in TR 23.701 which portrays telco interest in emergent web architectures. The solutions were described in *Section 4.4* in order to show the different architectural models where different components work in concert. Beforehand, the opening of the

chapter gave an overview of a basic integration architecture upon which all solutions could be viewed and interrogated according to the set of requirements that were detailed which were based on insights from the previous chapter.

The detailed description of each solution was important and necessary to the discussion in order to explicitly identify their similarities and differences. Through this process, it was discovered that Solution 1 represents a generic introduction of an RTCWeb inter-working function that is meant to express the need for having mediation functions for both signalling and media handling. These mediation functions were further described and split into subsequent solutions. The registration and session handling scenarios from Solutions 4, 5 and 6 were mainly based upon Solution 3 which describes the different modes of applying web authentication procedures to clients seeking to access IMS services through a web server or similar function managed either by the operator or third-party service provider. In addition, Solution 6 in particular, addressed the ability of the network to interact with another similar network domain where business trunking interfaces are established to connect them. Furthermore, Solution 7 provided the novel use case of using existing IMS assets such as SIMs to address the open issue of identity management for WebRTC, thus restricting use of web-based authentication schemes contrarily to the solutions before it. As a result, Solution 7 specified innovative ways of using a WWPF to provide the interface for the browser to access the user's IMS credentials retrieved by the IMS client located in an advanced and state-of-the-art UE.

Finally, *Sections 4.5, 4.6* and *4.7* provided further insights to be gained from the solutions analyses, the selection of the 3GPP reference architecture and how they lead to the potential to evolve IMS entities enhanced to natively support WebRTC. Even though the theme of the evolution of the IMS network contradicts a more general overall aim which is to prevent extensive modifications to either the WebRTC or the IMS network, there seems to be a clear necessity for such enhanced entities. However, the specification of an enhanced integration architecture is largely conceptual and there is a paucity of evidence of experimental trials. As a result, the next chapter will present the design and implementation of an alternative architecture which is motivated by this thesis in order to satisfy the requirements discussed thus far. The subsequent chapter will detail the open source tools and platforms that can be employed to realise an environment to investigate the integration.

5. Chapter 5 – The Integration of WebRTC and IMS: Proposed Model

5.1. Overview

The purpose of this chapter is to present the model that is espoused by the thesis. It aims to do so through an investigation of the requirements described in the previous chapters, and will contrast this model with the 3GPP reference architecture to explicitly highlight how certain design decisions were made that reflect the desired functionality according to the specific goals and scope of the thesis. That way, a novel integration model is expressed that is not simply a carbon copy of the 3GPP reference architecture, and more strongly accommodates the open source web / telco developer whose main goal is to experiment with an integrated architecture in a meaningful but cost effective way that is still strongly aligned with the spirit of the 3GPP recommendations.

5.2. Synthesising the Model

The model is derived to some extent from the 3GPP reference architecture, with its synthesis also borrowing certain aspects from the architectural alternatives that have been investigated and specified in the previous chapters. However, the model is novel in that it is constrained by the specific requirements that the thesis posits, with the aim of simplifying the design, and enabling an easier implementation path for the application developer who seeks to work in this area. Thus, the model is instrumental in communicating the ease of integration by emphasising the high-level differences that occur between it and the 3GPP solutions. In order to provide a more fluid discussion, as well as to simplify the process of highlighting the differences between this model and the other solutions, the discussion is structured in a like manner, with a discussion on general architecture, followed by an outline of registration and session handling scenarios.

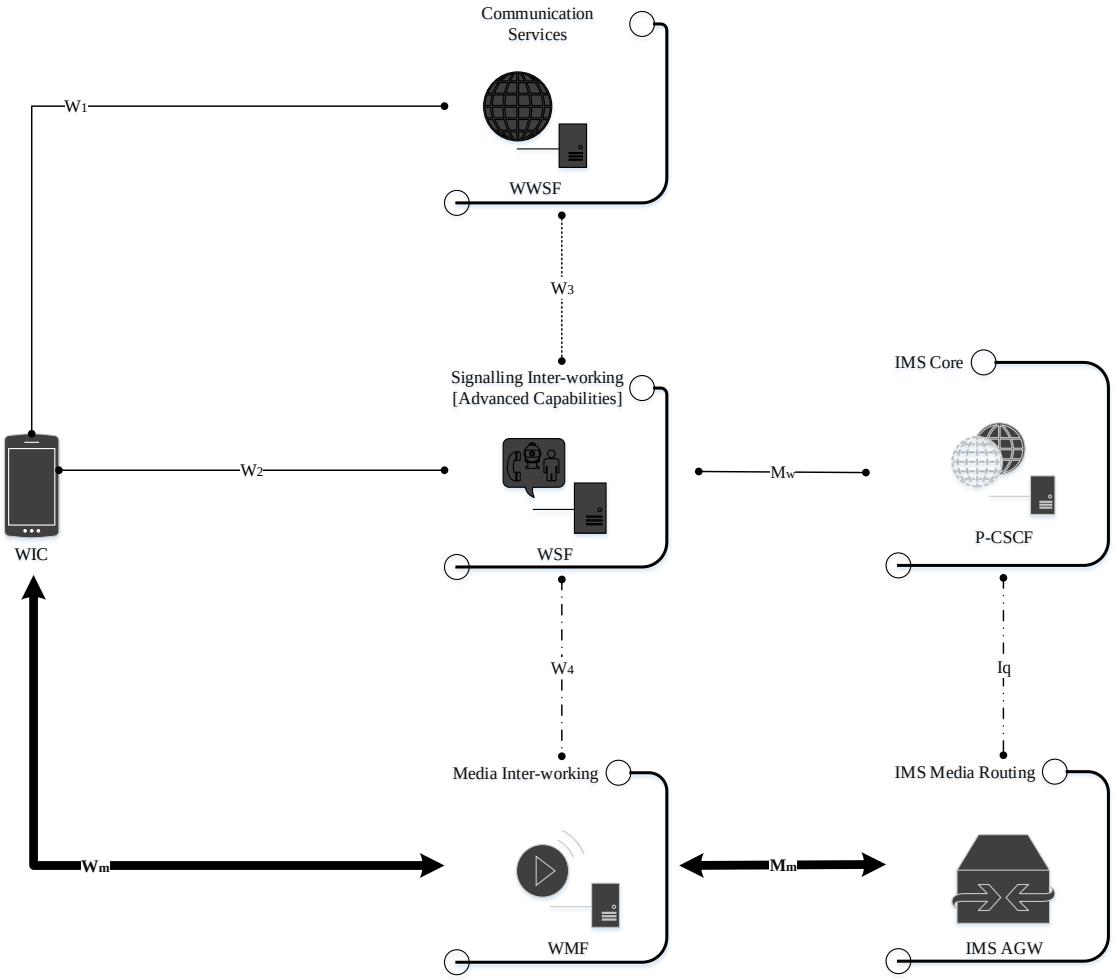

Figure 5-1 - WebRTC and IMS model.

5.2.1. Architecture

The model closely matches Solution 5, where the WIC is a browser-based or similar JavaScript execution environment acting as the black box that "provides application logic and WebRTC API calls to access to the communications services of the IMS" (Pascual, 2014). The sections below describe the functionality of each component in the architecture.

5.2.1.1. The WWSF

The model re-imagines the WWSF as a lightweight web server that simply hosts the WebRTC application and can work independently to implement a standalone service environment that provides a "plug-and-play" capability where advanced services such as identity management, contact management, third-party service integration, operations and business support systems and more can be implemented by supporting functions as needed for the application use case. These ancillary roles integrated into the WWSF can be recognised as dedicated components in the form of a WebRTC Portal / Unified Authorisation System, a WAAF and a WAF which have been demonstrated in Solutions 2 and 3, and the 3GPP reference architecture. The purpose of such an organisation of functions results in an architecture whose core value is relevance and innovation, thus meeting specific application and developer needs. Figure 5-2 depicts the WWSF and supporting functions.

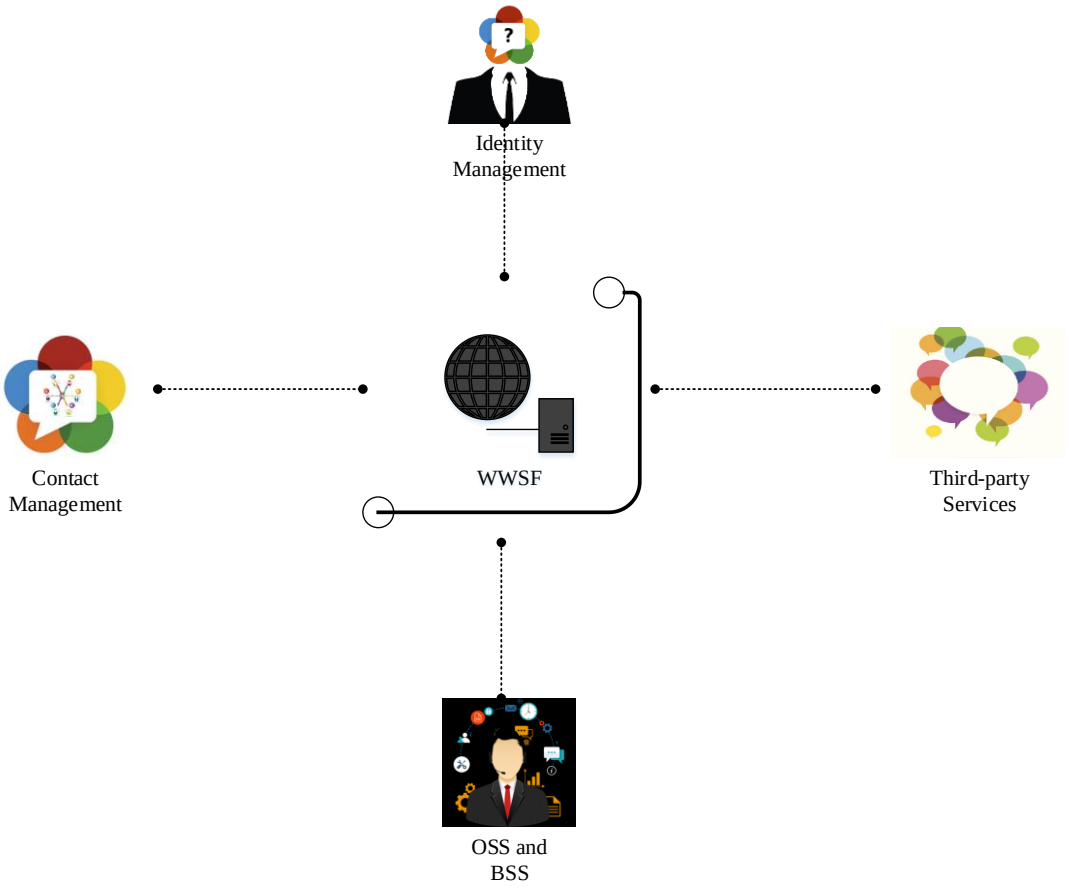

Figure 5-2 - The WWSF and supporting functions.

5.2.1.2. The eP-CSCF

The presence of the eP-CSCF is evolutionary in nature requiring major modifications to be made to the IMS core. The complexities surrounding these modifications results in technical implications for IMS, requiring it to support WebRTC frameworks, components, business models and value chains in addition to existing ones which are already criticised as being cumbersome. Considering this, it is expedient to deconstruct the eP-CSCF by arranging it in accordance with the structure proposed in Solution 4, where a WSF and the P-CSCF are either co-located or operate independently. Therefore, the WSF provides a lightweight signalling exchange mechanism responsible for translating SIP over WebSockets, with the ability to "plug in" support for additional signalling protocols, authentication schemes, third-party service control and others. The WSF also conforms to the WWSF "building block" service infrastructure where the P-CSCF is protected from undergoing extensive modifications outside of being able to recognise the new WebRTC access type. Figure 5-3 illustrates how the model should support additional signalling protocols.

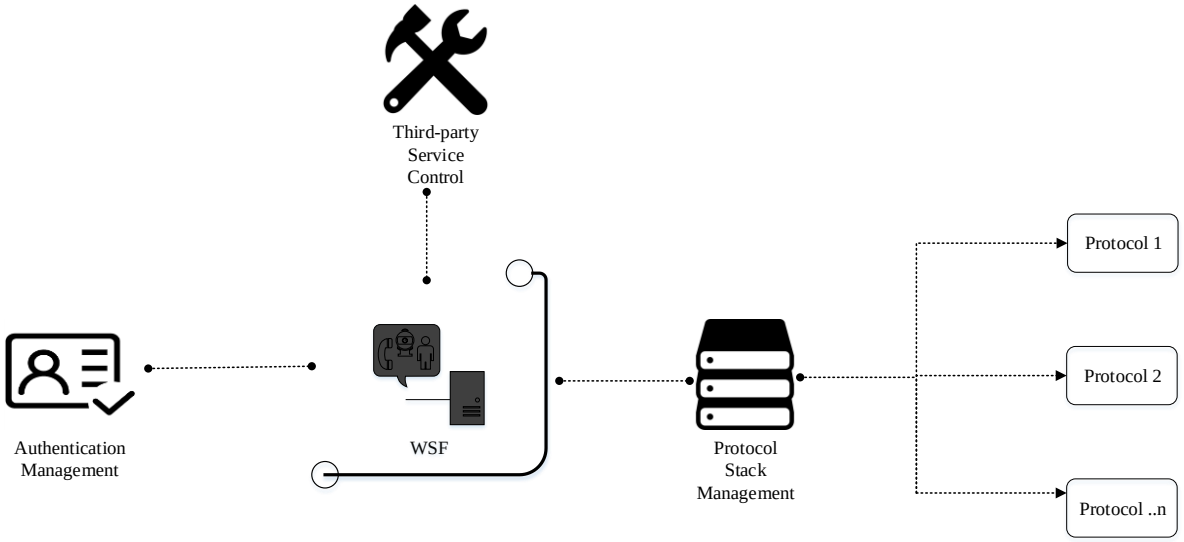

Figure 5-3 - The WSF and additional signalling supporting functions.

5.2.1.3. The eIMS AGW

The eIMS AGW combines a WMF and an IMS AGW to independently handle WebRTC and IMS media inter-working as demonstrated in Solutions 2, 4, 5, 6 and 7. The WMF is a standalone media environment, with the ability to support transcoding, DTLS-SRTP conversion and ICE connection management as basic use cases. When extended via supporting functions, analogous to previous clauses, the WMF has the potential to support use cases such as recording, voicemail, multiplexing, broadcasting, facial recognition and other advanced media processing capabilities. Figure 5-4 shows the WMF and some examples of media supporting functions.

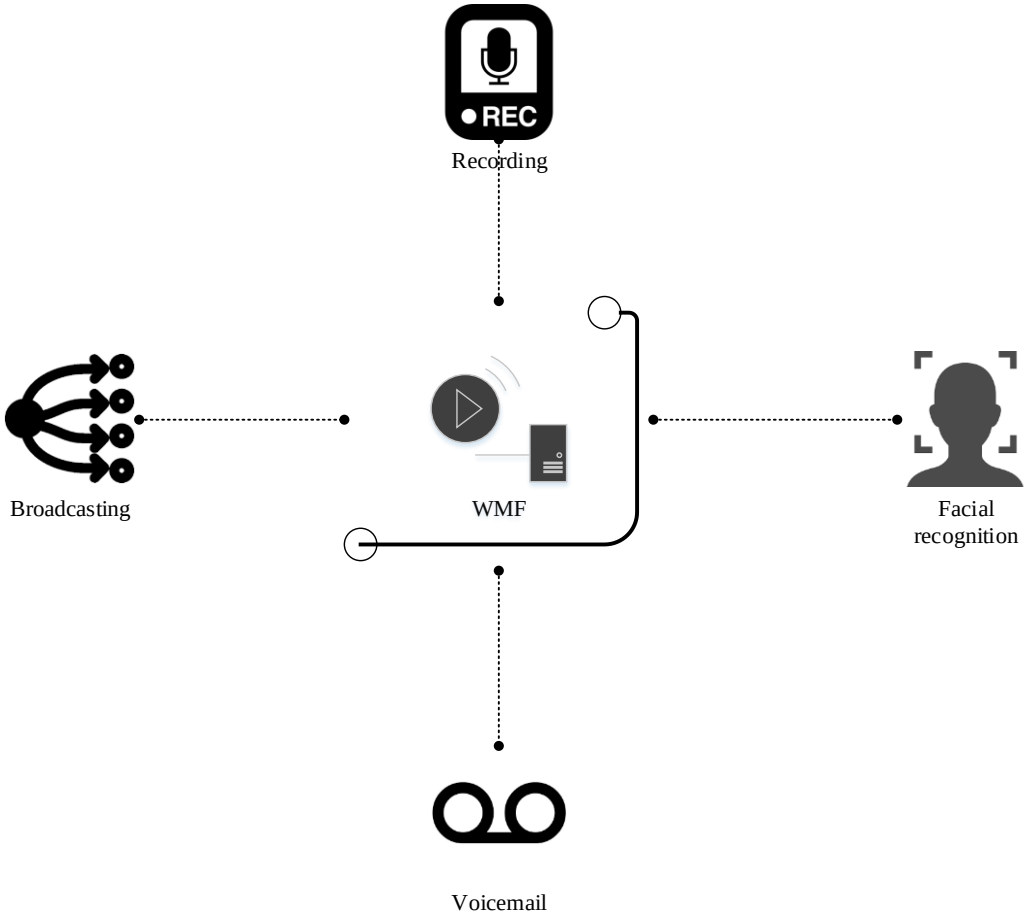

Figure 5-4 - The WMF and some supporting functions.

5.2.1.4. The WIC

The proposed WIC implementation is a modular, integrated system that is structured in such a way that the WWSF and the WIC can interact via an IMS proxy module in order to provide an interface to the supporting functions hosted by the WWSF. Examples of these supporting functions include an RCS Proxy, Operator SIM Authentication, Advanced Call Control, Notifications, Presence and Directory services and so on which have the capacity to implement certain API functionalities relevant to the deployment environment. The ability to access APIs more efficiently results in a greater degree of flexibility where developers can either create native or browser-based clients. For instance, a developer can create a native client over, Android or iOS that is able to execute RTC services via the RCS Proxy, or they can create a browser-based application with the ability to access SIM-based identity information via Operator SIM Authentication mechanisms. Still more, the Presence and Directory services function can enable efficient contact management where Operator Web IDs are used to describe user information. This gives the developer far greater freedom during application development, and thus has the potential to improve mobile support for WebRTC, which according to VoipSwitch (2014), enhances the ability to customise client applications. Figure 5-5 illustrates WIC functionality in the proposed model.

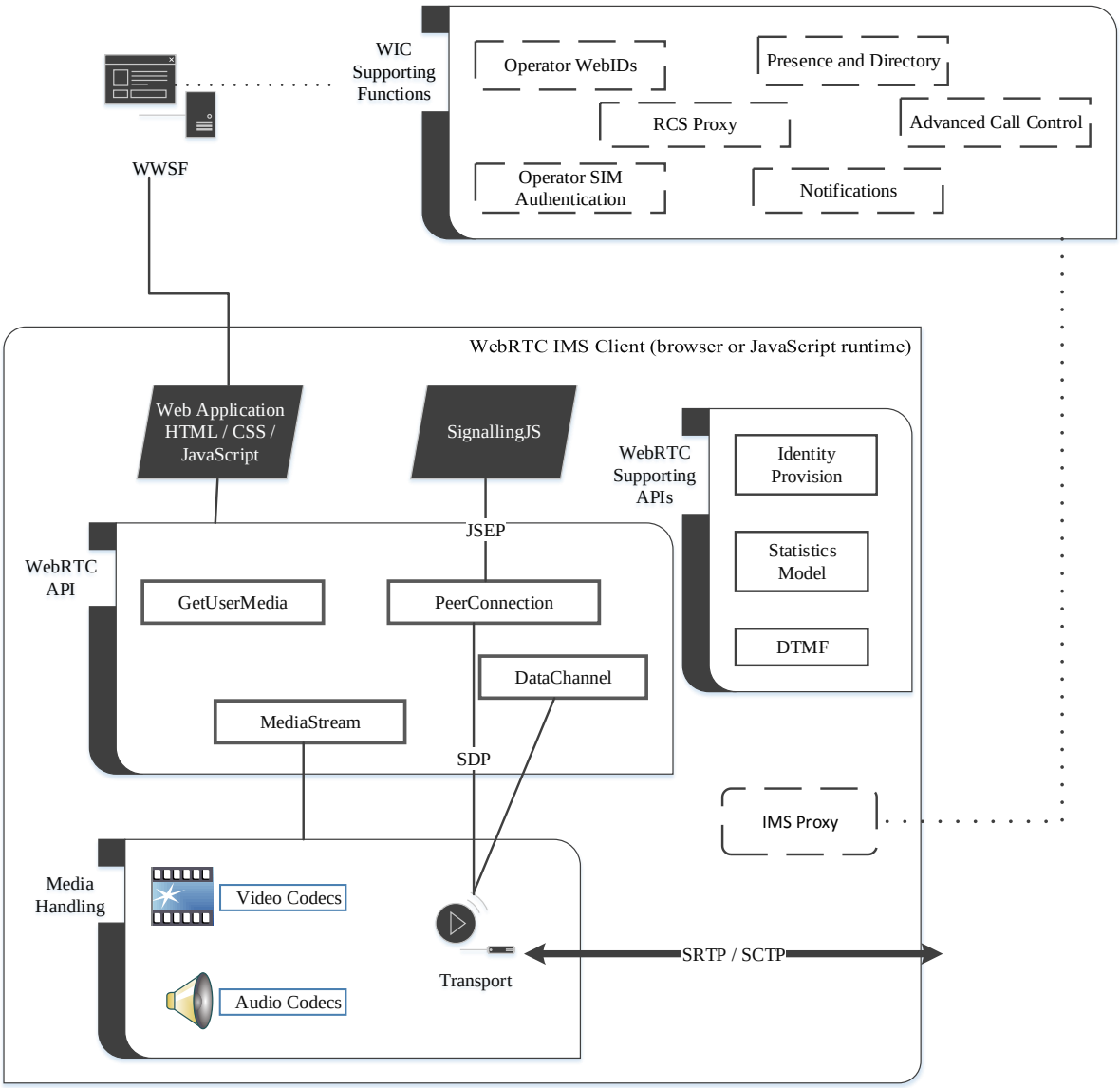

Figure 5-5 - The WebRTC IMS client architecture. Adapted from Taylor & Ing (2013).

5.2.2. Registration Scenario

The model supports call flows for registration and session handling scenarios that conform to those specified in the architectural solutions using SIP over WebSockets, however, these have been reimagined to include additional supporting functions. For instance, the call flow in Figure 5-6 illustrates a user registering their operator-provided web identity using XMLHttpRequest as the main communication channel, which is contrary to the preferred WebSockets method. The purpose of this call flow is to exemplify how the authentication management function could be invoked.

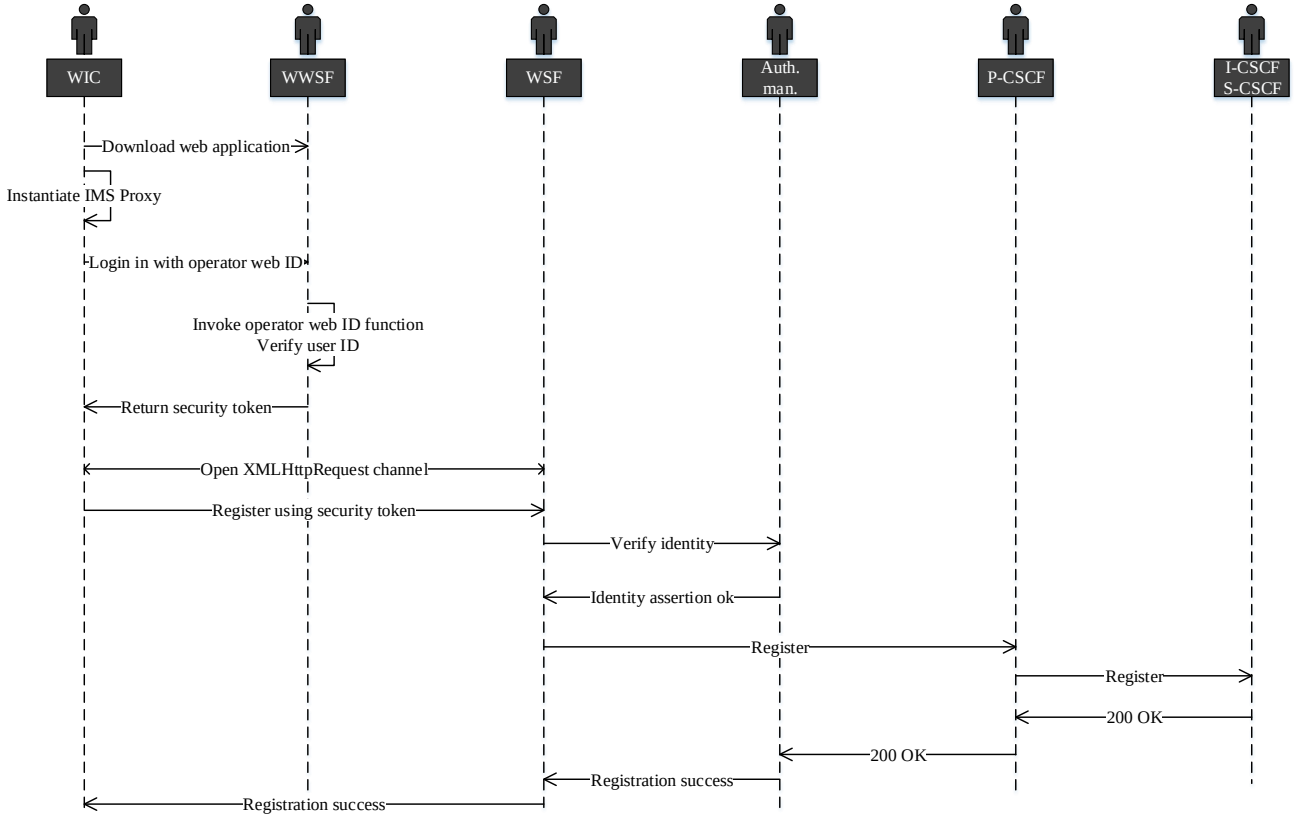

Figure 5-6 - Registration scenario using different signalling protocol and channel (JSON over XHR).

5.2.3. Session Handling Scenario

Once authenticated, the user may wish to use a messaging component of the application that is implemented as an RCS service where the RCS Proxy is invoked by the WWSF to handle the necessary inter-connection. With RCS reusing certain IMS functionality, the WSF is necessary to convert the WIC-side signalling to SIP. Following successful session establishment, messages can now flow between clients via the WMF and IMS AGW for protocol conversion and resource reservation respectively. The call flow in Figure 5-7 illustrates how the different functions could be used.

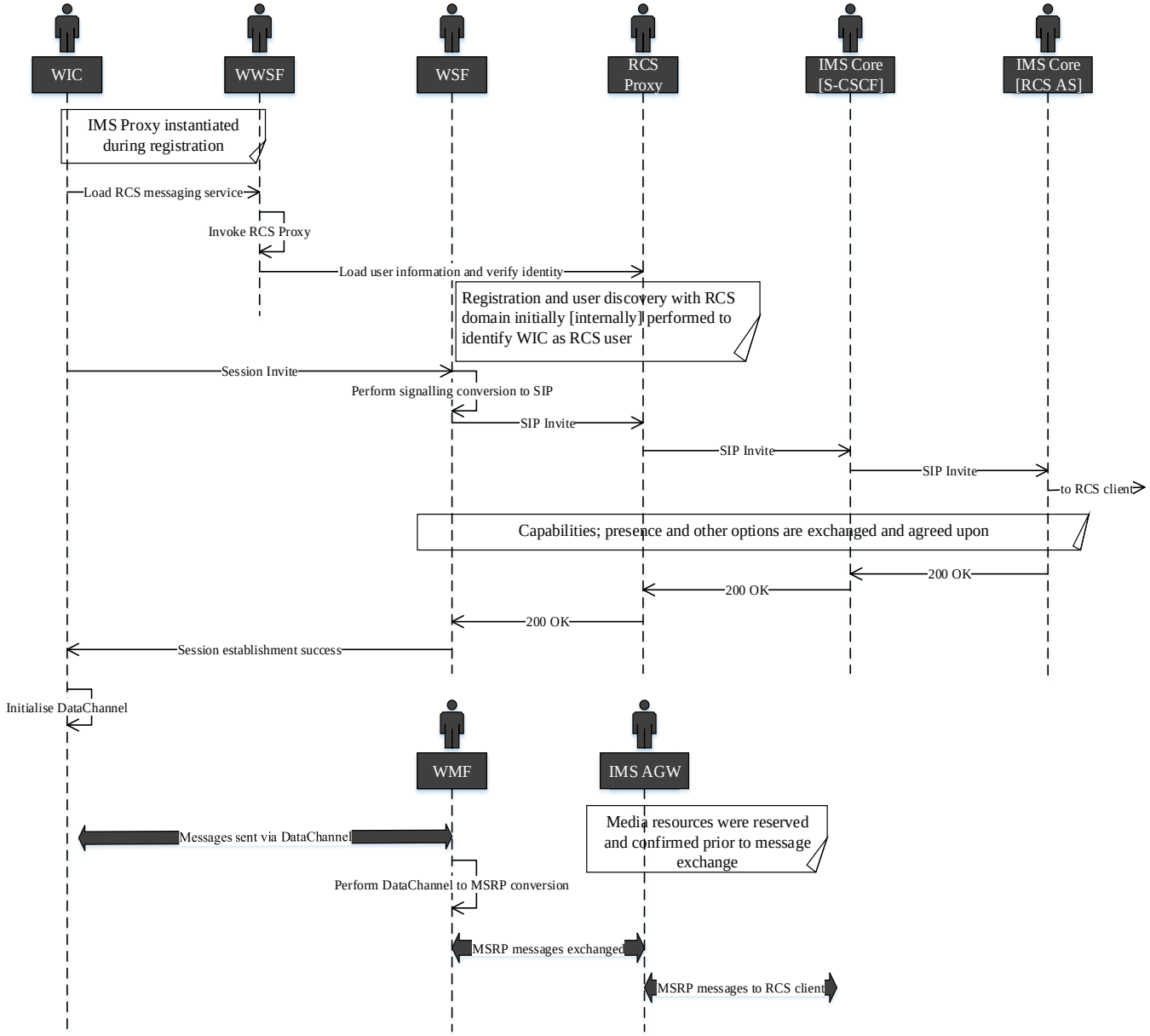

Figure 5-7 - Session handling scenario showing WIC using RCS messaging service.

5.3. Mapping the Model to the Requirements

The design considerations that were followed in the creation of the model are summarised in the points below which relate how the requirements organised in Table 3 are mapped to the arrangement of the different components included in the overall model. The discussion that follows identifies the way in which both the overall design and the different components satisfy the stated requirements.

REQUIREMENT	**DESCRIPTION**
	BASIC ARCHITECTURAL REQUIREMENTS
1	The use of gateway functions as bridges to external networks
2	The reuse of existing infrastructure
3	The use of internal and external protocols
3A	The implementation of internal IMS protocols
3B	The use of external protocols
	EMERGENT REQUIREMENTS FROM SOLUTIONS ANALYSIS
4	The ability to create integrated clients and UEs
4A	The ability to integrate with existing telco service platforms
4B	The ability to extend current UEs to the web domain
5	The potential to natively support web-based techniques
5A	The use of operator-based web identities
5B	The use of web-based identity protocols
5C	The ability to handle signalling alternatives
5D	The ability to handle WebRTC-based media

Table 3 - Requirements for the integration architecture.

1. The overall model

The overarching aim driving the creation of the model is the need to derive an architecture that is simple, effective and does not components that would not be readily available to the average developer when it comes to the implementation of it.

2. The WWSF

The WWSF conforms to *Requirement 1* as the re-imagined AS function that, in addition to hosting client applications, interfaces with IWFs or other ASs residing in a third-party domain. This interfacing is achieved through implementing appropriate interfaces between functions where SIP and the WebRTC API are basic examples of internal and external protocols that *Requirements 3a* and *3b* advocate. The WWSF also conforms to other requirements in the event of supporting advanced features, for instance, when interfacing with an RCS network, *Requirement 4a* discusses the resultant implications of such a connection.

3. The P-CSCF and WSF

The reuse of the P-CSCF as the initial point of contact with IMS conforms to *Requirement 2*, while its combination with the WSF enables the possibility to evolve IMS to natively support WebRTC functionality in conformance with the different parts composing *Requirement 5.*

4. The IMS AGW and WMF

The role played by the IMS AGW indicates the reuse of an IMS media gateway to enable transcoding, protocol conversion and overall media handling during the integration with WebRTC which aligns with *Requirements 1, 2* and *5d*. Furthermore, the addition of and integration with a WMF necessitates the implementation of internal protocols to interface to the WSF thus following *Requirement 3a*.

5. The WIC

The WIC architecture describes client behaviour that is versatile in its ability to interface with other functions through standardised interfaces, which is mainly in accordance with *Requirement 4*. For example, interoperability with RCS could result in an integrated UE that also provides WIC functionality as *Requirement 4a* details. In another instance, a WIC that supports Operator SIM Authentication could champion the creation of WebRTC-consuming user terminals that are described in *Requirement 4b*.

5.4. Conclusion

This chapter has described the synthesis of a WebRTC and IMS integration model whose definition was guided by the requirements that are unique to the present research. These requirements led to the construction of an architecture that addresses the design considerations of the 3GPP reference architecture and adapts it to a scope that is more relevant to the application developer as opposed to one that meets the needs of the operator. *Section 5.2* describes this model creation where the WWSF and WAF were firstly unpacked to show their conformity with *Requirement 1*. Secondly, *Section 5.2.1.2* saw the introduction of the WSF which was brought about from the decomposition of the eP-CSCF, hence incorporating aspects from *Requirements 2* and *5*. The decomposition of the eIMS AGW also followed a similar pattern where the WMF was introduced to work in conjunction with the IMS AGW to perform media inter-connection. Finally, the WIC architecture was detailed to show the interaction between the WebRTC API and other supporting functions that the operator could provide as API implementations. *Section 5.3* concluded with a mapping of the model components to the requirements, hence synthesising an integration model that presents numerous opportunities for experimentation. To this end, the next chapter illustrates the implementation of a network testbed that realises the model by presenting an array of open source tools that are currently readily available and can be used to aid the developer in this regard.

6. Chapter 6 –Prototyping the Model

6.1. Overview

The model presented in the previous chapter is a conceptual framework that demonstrates how to integrate WebRTC with IMS in such a way that the constraints and requirements of the present research are met. As such, it is important to verify the efficacy of the design with a practical implementation that satisfies the underlying requirements which demand a design that can be readily implemented and extended by the average developer. This chapter shows how the model was implemented using a selection of readily available, free and open source tools and platforms that are effective at mapping standards to practice.

6.2. Demonstrating the Model using Software Tools

There are a number of open source products that can be used to demonstrate the integration of WebRTC and IMS which have supported the creation of a practical environment for experimentation and the extension of the proposed model. This section therefore discusses the use of such tools to aid in the construction of a suitable network testbed. In particular, the investigation was strongly influenced by Loreto & Romano (2014) and Altanai (2014) who provide practical guidance in the development of WebRTC systems in general. As such, the discussion of the architecture involves a mapping of the software tools shown in Figure 6-1 to the model, where the process undergone to integrate each tool is also described. The registration and session handling scenarios are further described using call flows, code snippets and extracts from configuration files where necessary, in order to enhance the discussion.

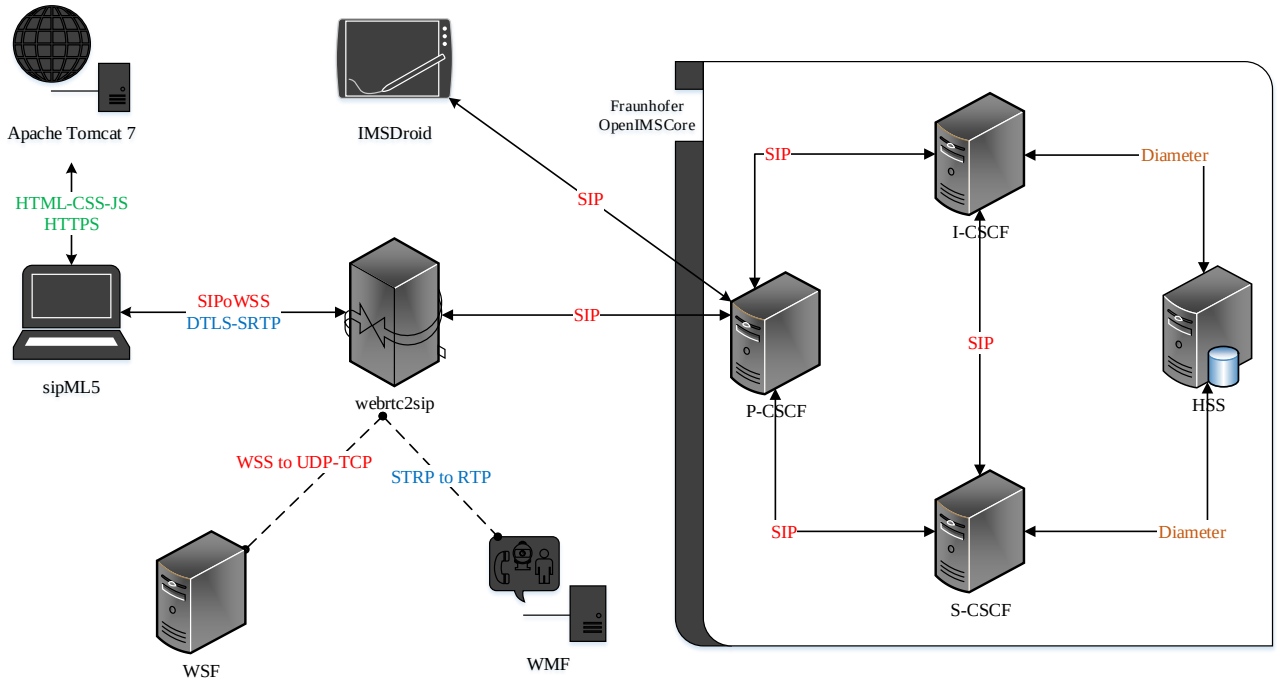

Figure 6-1 - Model demonstrated using software tools.

6.2.1. Hardware Platform and Environment Variables

The hardware platform used to execute the testbed comprises a Proline Officeware personal computer with the following hardware specifications:

- MSI X58 Pro-E (MS-7522) motherboard;
- Intel(R) Core(TM) i7 930 @ 2.8GHz CPU;
- 4GB RAM; 64-bit memory address size and
- ATA Disk size 500GB.

The network specifications include a 1 Gbit/s Ethernet interface attached to a personal computer with an Ubuntu 14.04.4 LTS operating system. The open source nature of Linux conformed to the research requirements, while the choice of Ubuntu was seamless due to the ease at which the environment could be customised and setup for development. The system environment variables are specified as follows:

- A private DNS server was configured using BIND9

The purpose of setting up a private DNS server was to ensure the effective management of services in the private network – the local DNS server acted as a DNS forwarder to an upstream DNS server for external Internet access. The use of a fully qualified domain name (FQDN) was preferable to using IP addresses in order to ease the maintenance of configuration files, applications and services. The domain chosen for the research was *webrtc-ims.co.za*.

- An SSL certificate was obtained to secure multimedia transports

A GeoTrust: RapidSSL® certificate was purchased through Register Domain SA in response to the move made by Google, as stated in Dutton (2015), to reject the implementation of the `GetUserMedia API` and media exchange via non-secure channels from Chrome version 47, a controversial move that also resulted in the failure of many self-signed certificates (Google Groups, 2015; Stackoverflow, 2016). However, the use of localhost is still enabled but unfortunately, unsuitable for most research needs, including the present.

6.2.2. Architecture

The discussion that follows provides an overview of the different options that were available, not only to provide a report on software alternatives, but also to justify the selection of the tools that were used in this implementation. SIP was the signalling protocol of choice for the demonstration because it is used pervasively in the IMS and therefore more easily supports the basic “barebones” architecture proposed for the model, hence the emphasis on SIP-based WebRTC tools and IMS registration and session handling procedures.

6.2.2.1. The WWSF: Apache Tomcat 7

The main requirement for the WWSF is to provide a lightweight web server function that has the ability to implement advanced features through additional functions. Apache Tomcat 7 was found to be appropriate at meeting this requirement, thus running as the HTTP web servlet container from which to execute the WIC. Tomcat is a project developed by the Apache Software Foundation in an effort to address the need “to simplify the creation of web applications” (Bakore, 2003) while also enabling support for integration with the Apache Web Server when extending the server architecture through modules. Tomcat is a Java-based web application container that runs servlets and Java Server Pages (JSP) in a stable, open source environment and is released under an Apache Software Licence version 2.0 (Vukotic & Goodwill, 2011). For this reason, it boasts a wide user-base and implements several

Java EE technologies, including support for WebSockets. Apache Tomcat versions are currently available as stable releases – the latest one at the time of writing being version 8.5.15, released May 5th 2017 with an alpha version (9.0.0.M21) released in May 4th 2017.

It is notable that Tomcat was the initial point of reference for the WWSF preferable over tools such as Microsoft Internet Information Services (IIS); Nginx and the Google Web Server (GWS) which, as stated in a survey conducted by Netcraft (2017), are the most used web servers on the Internet alongside Apache. The survey showed that Apache had the highest share of all sites that were assessed with a market share of about 45.8%, translating to almost 80 million active sites. Hence by extension, Tomcat was a highly rated choice. Furthermore, experience of the installation and setup of Tomcat in Ubuntu was seamless: OpenJDK Runtime Environment (IcedTea, 2.6.8) was used to run Java version 1.7.0_121; JAVA_HOME environment variables set and server started as a service over port 8085.

6.2.2.2. The WIC: sipML5

There are several JavaScript libraries available that can enable the creation of clients and user agents to provide SIP signalling for WebRTC applications. The main libraries/clients include jsSIP, a lightweight client-side library run over Node.js; SIP.js, a popular fork of the jsSIP library; sipML5, a feature-rich client developed by Doubango Telecom (2018) that can connect to SIP, IMS or PSTN networks; QoffeeSIP, a CoffeScript SIP stack for WebRTC; ctxPhone, a simple phone based on SIP.js, and many others. These clients were tested individually for the purpose of comparison, but ultimately, jsSIP and sipML5 were chosen as the main solutions for the testbed. Their selection was based on the manner in which they are packaged which is as part of a comprehensive gateway function that can easily be extended or manipulated to meet a number of mediation needs, and thus be more adept at addressing the integration needs of the research.

jsSIP was developed as the testing library for OverSIP, an outbound SIP proxy server, and can work with other popular SIP servers such as Kamailio, Asterisk, OfficeSIP and others that support WebSockets. Similarly, sipML5 was developed as the main client to work with webrtc2sip, a gateway developed by Doubango Telecom (2016b) as a software artefact to enable WebRTC endpoints to communicate with legacy SIP and PSTN networks. Even though jsSIP is better maintained, with more recent releases, it is the ability for the sipML5 and webrtc2sip package to interoperate particularly with IMS that resulted in their selection above the other tools. As such, the next section describes the sipML5 architecture. webrtc2sip is described in *Section 6.2.2.4*.

The sipML5 architecture

Altanai (2014) demonstrates three ways of using sipML5: the most basic option is to use the demo version available online at http://sipml5.org/call.htm. The second option enables a developer to code a simplified version of the sipML API, and use a basic web server (or in this case a web servlet container) to load it over a WebRTC-enabled browser. Finally, the third option is to access the sipML5 source code that can be checked out from GitHub and modified for greater customisation – this option is most suitable for developers looking to integrate sipML5 over other systems and was therefore followed for this research (sipML5, 2017).

During the development phase, version 1.5.222 was available and is the one that was eventually used, however, at the time of writing, version 2.1.3 is available and regularly updated to fix interoperability issues with Asterisk, a software PBX. Figure 6-2 below shows the graphical interface that enables users to configure their accounts and Figure 6-3 shows the graphical interface for users to configure settings under 'Expert Mode' with information about WebSocket, SIP and ICE servers - the media handling

settings are also configured on this interface. The screenshots show the network settings for a user account served over the *webrtc-ims.co.za* domain.

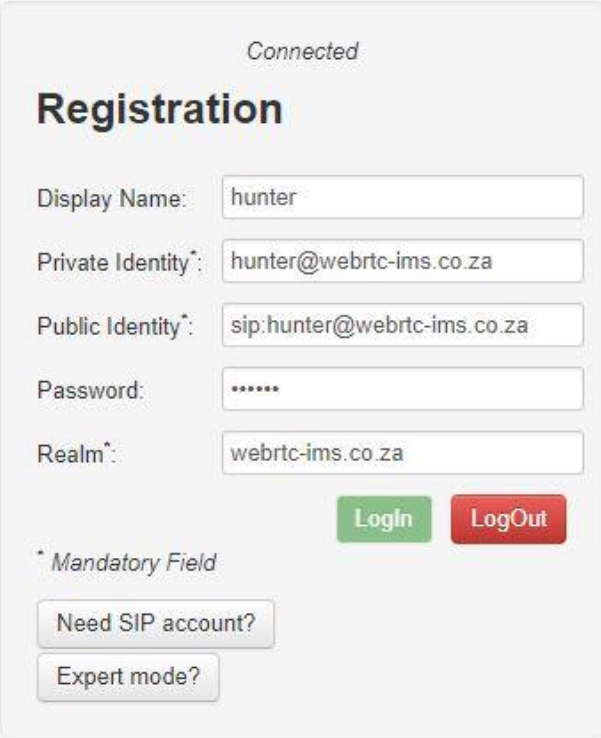

Figure 6-2 - Registration interface on sipML5.

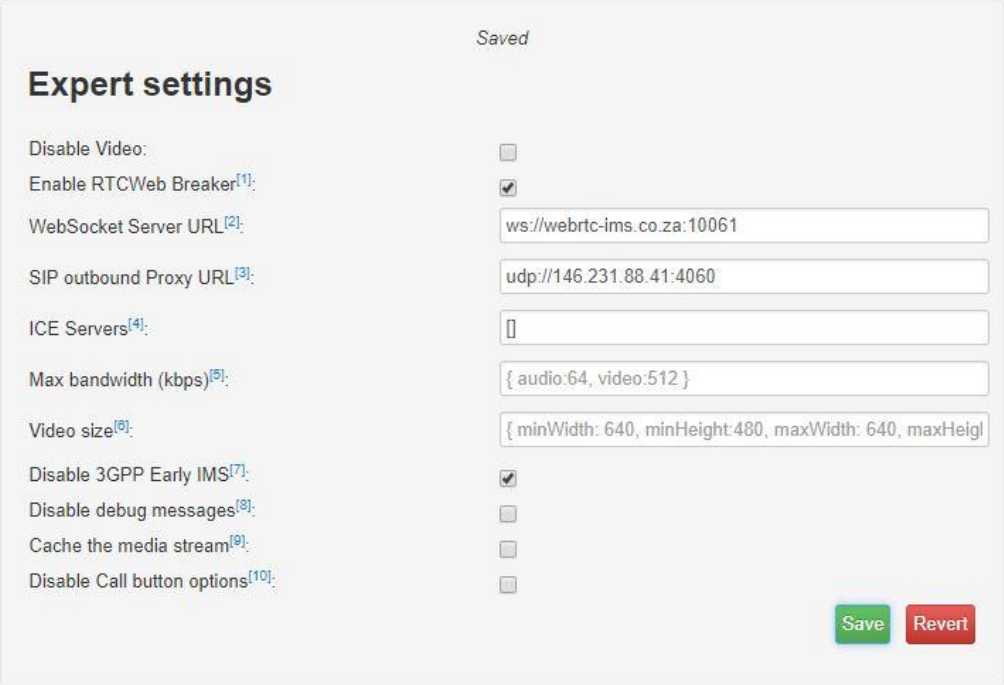

Figure 6-3 - Interface to configure network settings by experts on sipML5.

6.2.2.3. The IMS Client: IMSDroid

Work carried out by Segec & Kovacikova (2012), Spiers & Ventura (2010), El Alaoui et al. (2012) and others served as starting point towards reviewing the IMS client landscape, which includes a number

of clients, each with its own features and target platforms. For instance, the Boghe IMS/RCS client was developed for Windows, iDoubs by Doubango Telecom for iOS, IMSDroid for Android, the UCT IMS Client for Linux platforms and others such as myMonster TCS, IMS Communicator and so on. Client functionality covers a wide variety of use cases, from basic use cases like registration, one-to-one voice and video calls and instant messaging; to more advanced ones like presence, IPTV, contact management, advanced authentication schemes (security) and more. The process of testing these clients revealed that some of them are not being actively maintained and rely on outdated and sometimes buggy software libraries, thus making their integration within the testbed a challenge. For instance, the UCT IMS Client, myMonster TCS and IMS Communicator have not been updated since 2014. In the end, IMSDroid was deemed the most suitable client to use in the demonstration of the model. Figure 6-4 and Figure 6-5 show the client's interface showing provisioned user and network settings.

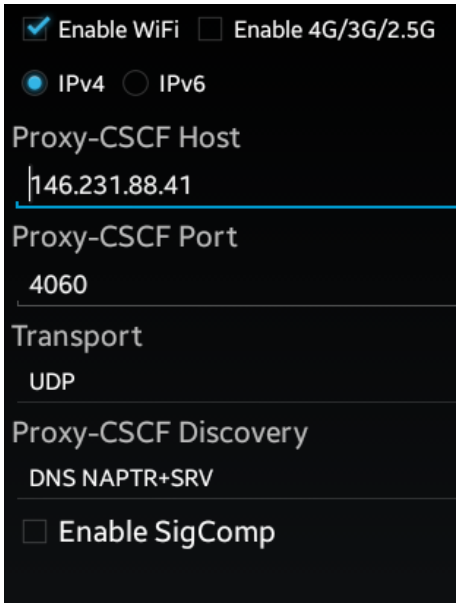

Figure 6-4 - IMSDroid network details.

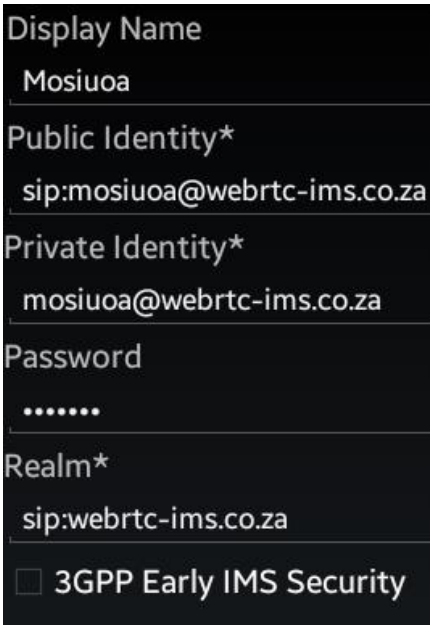

Figure 6-5 - IMSDroid user account details.

6.2.2.4. The WSF and WMF: webrtc2sip gateway

The webrtc2sip gateway was developed as a software artefact to enable WebRTC endpoints to interact with legacy SIP and PSTN networks. An important observation to make through the inclusion of the gateway is the removal of the need to provide an explicit IMS AGW to handle media processing, as the gateway is equipped to do so. However, the research recognises the importance of this entity therefore recommendations are made in *Section 6.3* for a likely tool to use that could act as the IMS AGW for the benefit of controlling IMS-side media. The gateway was the initial and most preferred tool for testing due to the feature-rich capabilities it offers which include the combination of the WSF and WMF roles. It has a modular architecture that comprises a SIP proxy to convert WebSockets to UDP, TCP or TLS; an RTCWeb Breaker, to convert DTLS-SRTP media to RTP/RTCP and to negotiate media flow using ICE and a Media Coder to translate audio and video codecs accordingly. These modules can be classified within the model as follows: the SIP Proxy as the WSF, and the RTCWeb Breaker and Media Coder as the WMF. Furthermore, it includes a Click-to-Call service for service providers to use on social media profiles and company websites. Version 2.7.0 of the software was used and is freely available under a GNU General Public License (GPLv3), which permits users to freely access, modify and distribute the software so long as derivative work is also available under the same license (GNU, 2007). Figure 6-6 shows the webrtc2sip architecture.

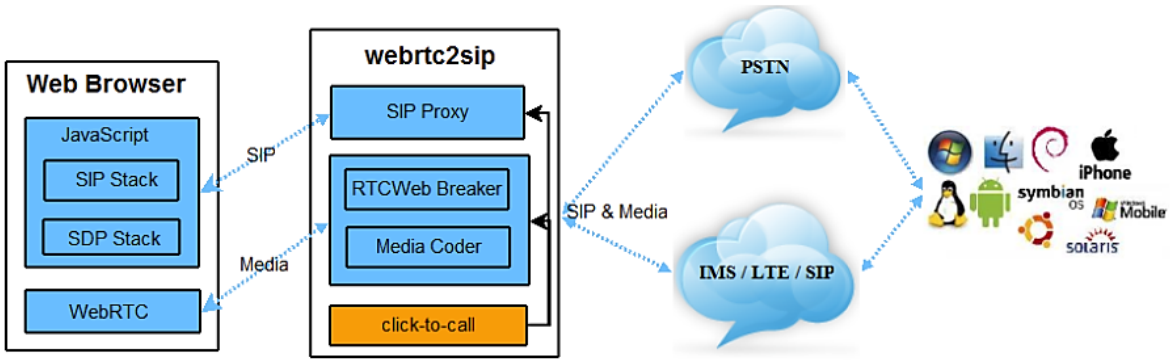

Figure 6-6 - The webrtc2sip gateway architecture. Source: Doubango Telecom (2016b).

The configuration of webrtc2sip was carried out in a configuration file named *config.xml* file which was configured with the contents shown in Figure 6-7:

```
<debug-level>INFO</debug-level>

<transport>udp;*;10060</transport>
<transport>ws;*;10061</transport>
<transport>wss;*;10062</transport>
<!--transport>tcp;*;10063</transport-->
<!--transport>tls;*;10064</transport-->

<enable-rtp-symetric>yes</enable-rtp-symetric>
<enable-100rel>no</enable-100rel>
<enable-media-coder>yes</enable-media-coder>
<enable-videojb>yes</enable-videojb>
<video-size-pref>vga</video-size-pref>
<rtp-buffsize>65535</rtp-buffsize>
<avpf-tail-length>100;400</avpf-tail-length>
<srtp-mode>optional</srtp-mode>
<srtp-type>sdes;dtls</srtp-type>
<dtmf-type>rfc4733</dtmf-type>

<codecs>opus;pcma;pcmu;gsm;vp8;h264-bp;h264-mp;h263;h263+</codecs>
<codec-opus-maxrates>48000;48000</codec-opus-maxrates>

<stun-server>stun.l.google.com;19302;*;*</stun-server>
<enable-icestun>no</enable-icestun>

<max-fds>-1</max-fds>

<nameserver>146.231.88.41</nameserver>

<ssl-certificates>
  /home/valerie-webrtc/domainCerts/webrtc-ims.co.za.key;
  /home/valerie-webrtc/domainCerts/webrtc-ims.co.za.crt;
  *;
  no
</ssl-certificates>
```

Figure 6-7 - webrtc2sip config.xml.

The points below highlight the relevant aspects to note regarding the contents of this file:

i. Transport variables

These fields specify where webrtc2sip is listening for incoming WebSocket connections, either ws (WebSocket) or wss (secure WebSocket), which are also stipulated in the sipML5 expert settings page.

ii. Media settings

The details concerning the media transport are stipulated from within the `enable-rtp-symetric` tag to the `dtmf-type` tag. The `enable-media-coder` tag, when set to 'yes', utilises the RTCWeb Breaker to convert and transcode media according to the media codec variables set in the relevant tags. On the client side, sipML5 checks the 'Enable RTCWeb Breaker' to enable the connection with IMS.

iii. SSL certificates

This section specifies the certificates to use for secure WebSockets and indicates the verify-value which when set to 'no', disables the validation process to ensure that the connection is established should the remote peer's certificates be missing or have a mismatch.

6.2.2.5. The IMS Core: OpenIMSCore

The OpenIMSCore was developed by the Fraunhofer FOKUS Institute in 2004 and in 2006 was released as an open source (GPLv2 licence) tool for IMS testbeds compliant with 3GPP and 3GPP2 standards (openimscore.org, 2015). According to Segec & Kovacikova (2012), the purpose of the project was to develop an advanced learning environment upon which complex multimedia experiments could be conducted to simulate real-world scenarios and also to ensure interoperability testing with other network components. Being the first of its kind, OpenIMSCore was an initial point of reference for the deployment of the IMS core network for this research, comprising CSCFs (P-CSCF; I-CSCF and S-CSCF) as modules; a lightweight HSS called FOKUS Home Subscriber Server (FHoSS) which is written in Java and employs a MySQL database for storage. The project also implements a CDiameterPeer module, also written in Java, for the Diameter stack which defines three interfaces: Sh, for ASs to access the HSS; Zh and Cx, to communicate with the I-CSCF and the S-CSCF.

For this research, OpenIMSCore was configured to run over the *webrtc-ims.co.za* domain and the default port numbers kept as follows: P-CSCF: 4060; I-CSCF: 5060; S-CSCF: 6060, and the web interface supported by a Tomcat container exposing the HSS running over 8080 and accessible from the client machine. User identities and network settings can be created and managed either via the web-based HSS management console (from the client machine) or the `mysql` server instance accessible from the command line.

From 2015, the project management of OpenIMSCore was taken up by Core Network Dynamics, a German start-up company that provides software solutions for mobile network infrastructure based on the Evolved Packet Core (EPC). Core Network Dynamics investigates 4G networks, with the intent to commercialise software-based end products (Core Network Dynamics, 2017). It is however, worth noting that an independent project that is not aligned with Core Network Dynamics exists in the form of Kamailio IMS, which extends the open source Kamailio SIP server to implement IMS functions (openimscore.org, 2015). Even so, the OpenIMSCore platform ran successfully in the network testbed system due to relevant documentation and online support platforms that make it easy to integrate as compared with Kamailio IMS, which is discussed in *Section 6.3*.

6.2.3. Registration scenario

The OpenIMSCore follows the standard IMS procedures for registration, therefore to avoid unnecessary duplication, this discussion focuses mainly on the SIP over WebSockets, abbreviated as SIPoWS where the "o" stands for "over", portion of the communication between sipML5 and webrtc2sip in order to highlight the protocol conversion that occurs. The call flow below describes the

registration procedure that employs the tools arranged in Figure 6-1, while Figure 6-9 and Figure 6-10 illustrate the conversion process using a screenshot of a captured SIP "REGISTER" message.

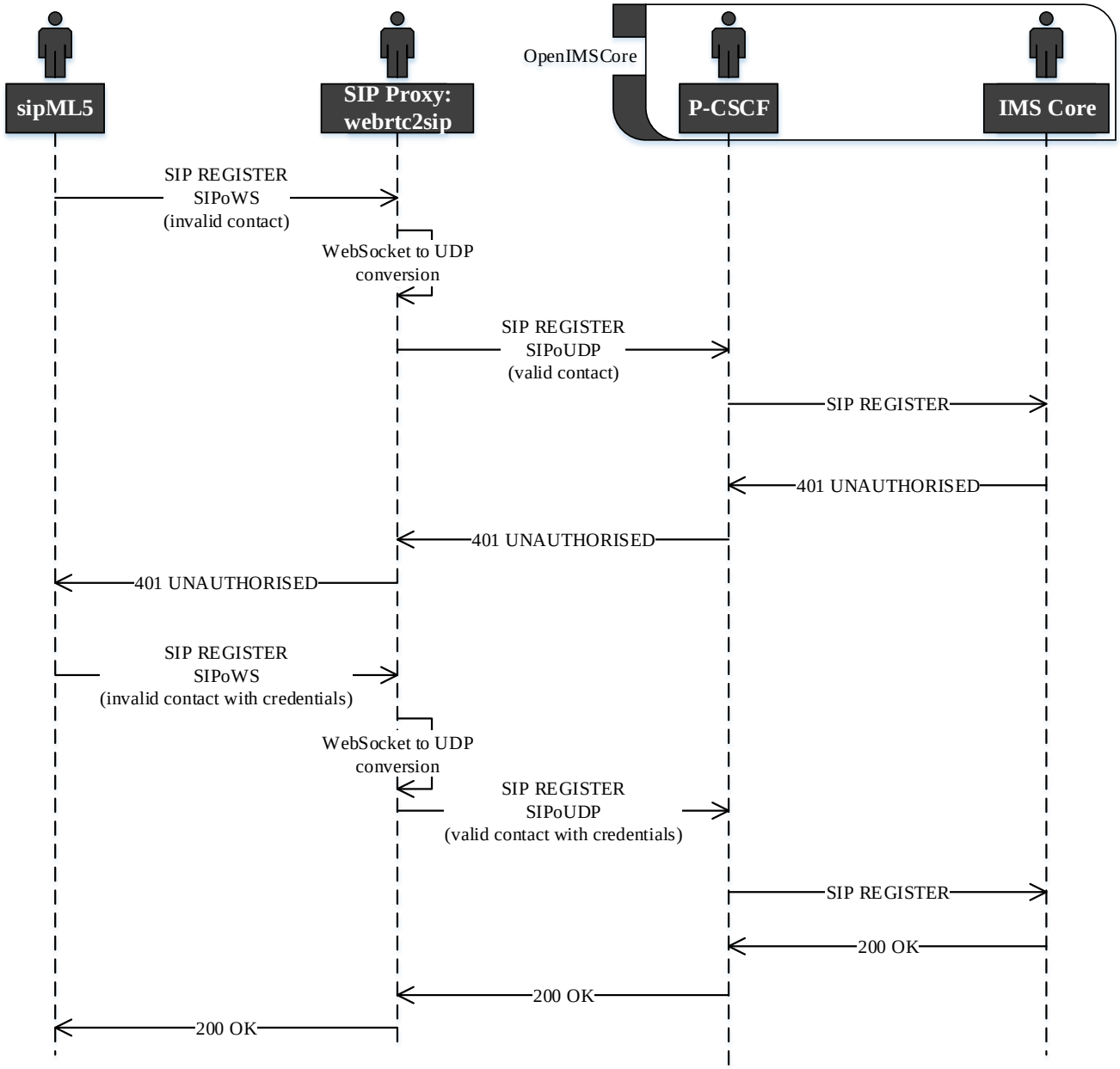

Figure 6-8 - sipML5 - webrtc2sip - OpenIMSCore registration scenario.

As the purpose of a SIP "REGISTER" is to form an association/binding between the user's AoR (for example, sip:hunter@webrtc-ims.co.za) and their IP address, port number and chosen transport protocol, the IMS needs to be able to capture this information in order to facilitate the registration. However, the browser is not able to retrieve and include it in the Contact and Via headers for IMS registration, as such, these headers are populated with a default invalid address. Furthermore, the use of WebSockets prevents the SIP network from handling the request, in the event that the SIP servers do not support WebSockets. As such, webrtc2sip examines the request and determines the IP address of the client and also translates the transport from WebSockets to UDP/TCP, represented as SIPoUDP in the diagram where the "o" also stands for "over".

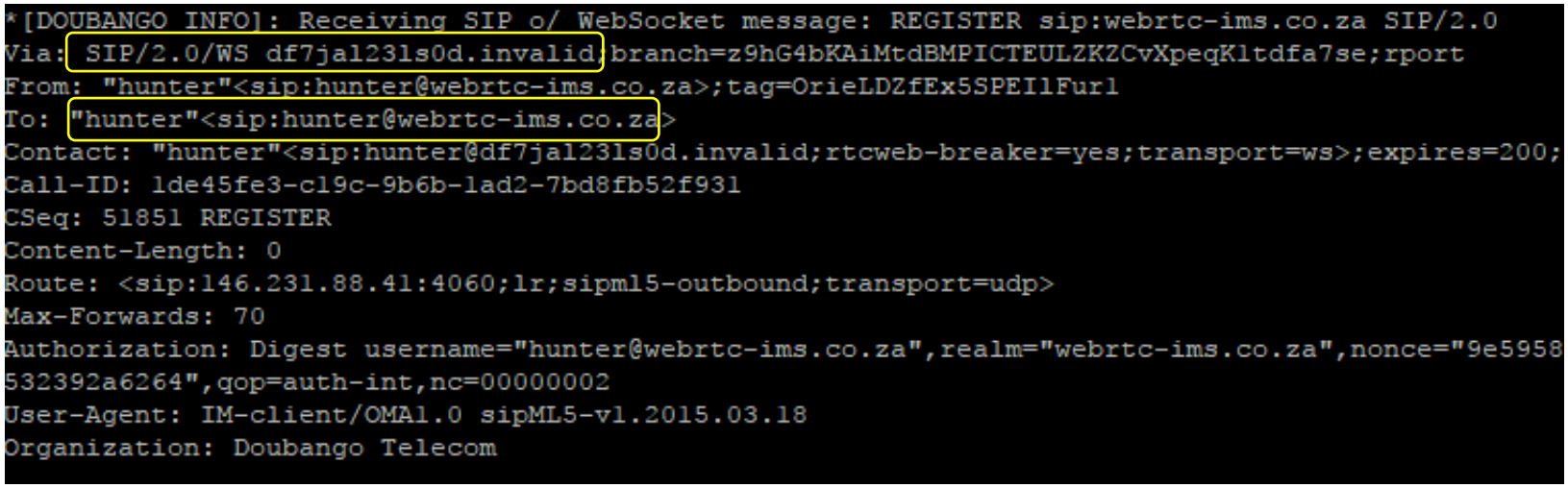

Figure 6-9 - webrtc2sip: local address retrieval.

The Via header is modified to use TCP as a transport protocol, and to include the IP address and port number pair from which the WebSocket connection was established and the request was received, in this case, the address of the client machine. The SIP Proxy module also adds an additional Via header to the request to indicate that the message traversed it, via UDP.

Figure 6-10 - webrtc2sip: transport conversion and updated contact header.

Once the request has been processed, OpenIMSCore receives the request in its appropriate format and challenges the user to authenticate via a 401 "UNAUTHORISED" response. The subsequent "REGISTER" message that sipML5 sends back to the OpenIMSCore with the credentials requires transport conversion to take place once more, following which, successful user registration can be confirmed via a 200 OK response as shown in Figure 6-11.

Figure 6-11 - webrtc2sip: 200 OK successful response from IMS.

6.2.4. Session handling scenario

Figure 6-12 shows a call flow diagram illustrating the session handling scenario where signalling and media traverses webrtc2sip from IMSDroid to sipML5.

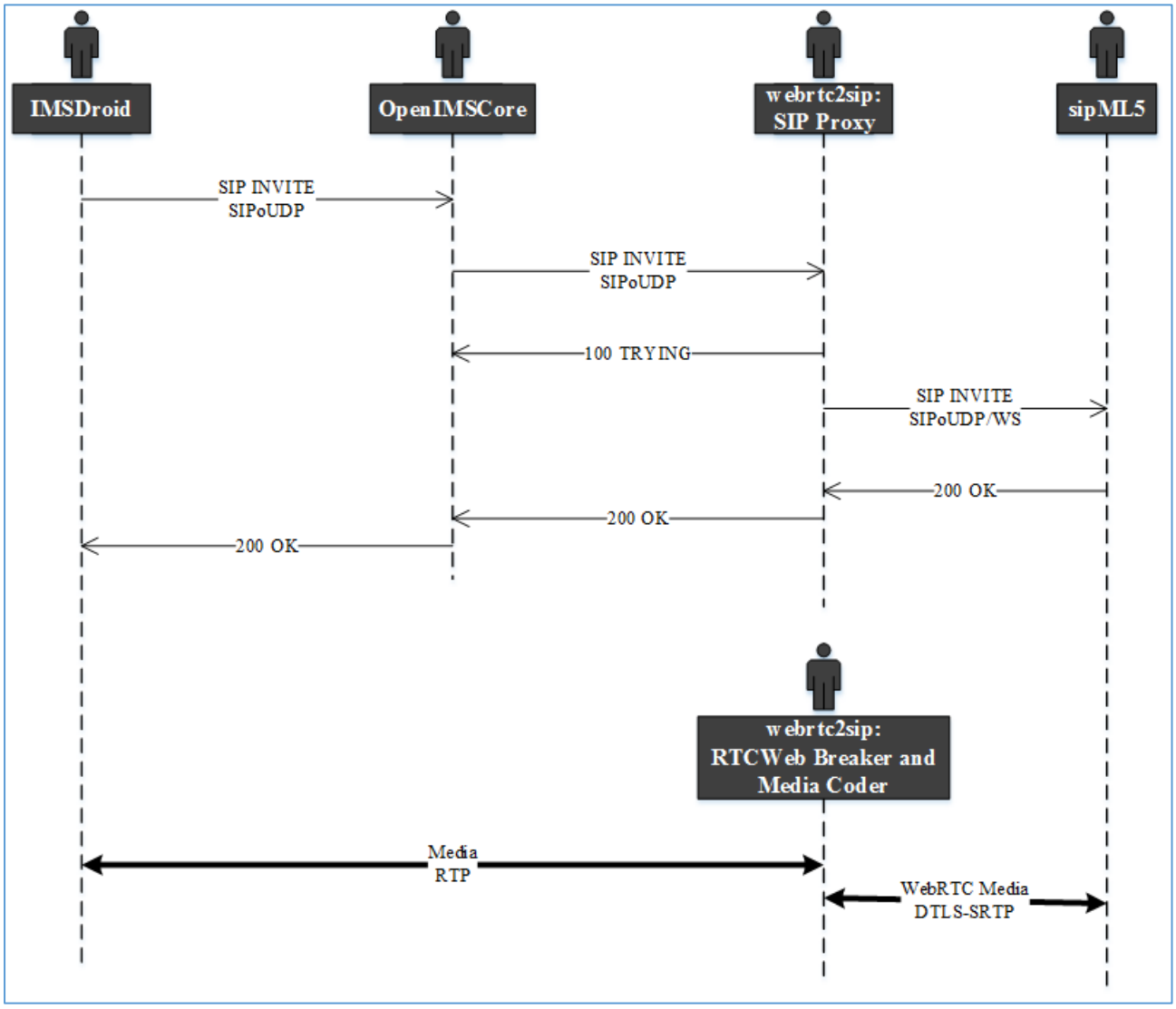

Figure 6-12 - Session handling scenario: IMSDroid - sipML5.

The SIP "INVITE" request is sent over the WebSocket connection and uses an SDP stack written in JavaScript to negotiate the media parameters. The request is also modified by webrtc2sip to translate the relevant Via and Contact headers with the appropriate source and destination contact addresses. Once media parameters are agreed upon, media can flow between clients via the webrtc2sip Media Server as shown in Figure 6-13 Figure 6-14 which show a screenshot of the SIP "INVITE" and SDP messages sent when users Mosiuoa and Hunter establish an audio session.

```
Via: SIP/2.0/WS 146.231.88.41:10061;rport;branch=z9hG4bK-4244686176
From: <sip:mosiuoa@webrtc-ims.co.za>;tag=4285534354
To: <sip:hunter@webrtc-ims.co.za>
Contact: <sip:mosiuoa@146.231.88.41:10061;transport=ws>
Call-ID: 3061cafa-1775-bdf9-3220-0141df298689
CSeq: 124240287 INVITE
Content-Type: application/sdp
Content-Length: 1475
Max-Forwards: 70
Route: <sip:146.231.89.134:59699;transport=ws;lr>
User-Agent: webrtc2sip Media Server 2.7.0
```

Figure 6-13 – SIP "INVITE" request during session handling scenario.

```
v=0
o=doubango 1983 678902 IN IP4 146.231.88.41
s=-
c=IN IP4 146.231.88.41
t=0 0
a=acap:1 setup:actpass
a=tcap:1 UDP/TLS/RTP/SAVPF UDP/TLS/RTP/SAVP RTP/SAVPF RTP/SAVP RTP/AVPF
a=acap:4 fingerprint:sha-1 69:1D:F1:28:9C:CB:BA:77:F1:1F:63:5F:4B:35:77:07:56:FF:8C:D2
a=acap:3 fingerprint:sha-256 DF:2A:D0:B9:0B:5B:79:B3:52:25:96:95:7D:43:47:C0:11:DD:FD:57:25:E9:C6:71:33:43:EF:28:00:14:34:EB
a=acap:1 setup:actpass
m=audio 40260 RTP/AVP 111 8 0 101
c=IN IP4 146.231.88.41
a=ptime:20
a=minptime:1
a=maxptime:255
a=silenceSupp:off - - - -
a=rtpmap:111 opus/48000/2
a=fmtp:111 maxplaybackrate=48000; sprop-maxcapturerate=48000; stereo=0; sprop-stereo=0; useinbandfec=0; usedtx=0
a=rtpmap:8 PCMA/8000/1
a=rtpmap:0 PCMU/8000/1
a=rtpmap:101 telephone-event/8000/1
a=fmtp:101 0-16
a=acap:5 crypto:1 AES_CM_128_HMAC_SHA1_80 inline:1SEA1FBMB64qPuLk85R2CWYDYYfmGb46eAFBkNVy
a=acap:6 crypto:2 AES_CM_128_HMAC_SHA1_32 inline:cVHlbJiTU/k4gLQEr5BRB6sEP+8020U053iU3DUh
a=pcfg:1 t=1 a=1,2,4|3
a=pcfg:2 t=2 a=1,2,4|3
a=pcfg:3 t=3 a=5,6
a=pcfg:4 t=4 a=5,6
a=pcfg:5 t=5
a=sendrecv
a=rtcp-mux
a=ssrc:3758843068 cname:(null)
a=ssrc:3758843068 mslabel:6994f7d1-6ce9-4fbd-acfd-84e5131ca2e2
a=ssrc:3758843068 label:doubango@audio
a=ice-ufrag:ya9KfCrWJ640TgB
a=ice-pwd:hZahJjn1JGrvwKwIRrKn2x
a=candidate:GdL3vOHrLl74Byg 1 udp 2130706431 146.231.88.41 40260 typ host tr udp fd 25
a=candidate:GdL3vOHrLl74Byg 2 udp 2130706430 146.231.88.41 40261 typ host tr udp fd 26
```

Figure 6-14 - Example SDP offer.

Table 4 below summarises the IP addresses and port numbers of all clients and services running in the testbed.

Client / service	**IP address**	**Port number**
P-CSCF: pcscf.webrtc-ims.co.za	146.231.88.41	4060
I-CSCF: icscf.webrtc-ims.co.za	146.231.88.41	5060
S-CSCF: scscf.webrtc-ims.co.za	146.231.88.41	6060
HSS: hss.webrtc-ims.co.za	146.231.88.41	3868 8080 (GUI management console)
Webrtc2sip gateway	146.231.88.41	10060 (UDP) 10061 (WS) 10062 (WSS)
sipML5 (WebRTC client)	146.231.89.134	8085 (Tomcat)
IMSDroid (IMS client)	146.231.183.95	48863

Table 4 - IP addresses and port numbers of clients and services.

6.2.5. Challenges

The main challenges facing the integration mostly involved outdated libraries or documentation, interoperability issues when integrating the different tools and lack or delayed support from the open source community. With WebRTC still an emergent technology, further challenges were experienced because browser developers introduced updates and changes to the WebRTC ecosystem. These challenges unfortunately frustrated the task of demonstrating the efficacy of the design through practical experimentation.

6.2.5.1. Code updates: `navigator.getusermedia()`

One of the frustrations experienced during the initial stages of the development of the prototype, in late 2015, included the deprecation of the `navigator.getUserMedia()` method to access the `getUserMedia` API (Mozilla, 2017). Previously, the API was prefixed with Webkit for Chrome (becoming `webkitGetUserMedia`); moz for Firefox (becoming `mozGetUserMedia`) and remained as was for Opera. Although the old method is still included in specifications, the newer method, `navigator.mediaDevices.getUserMedia()`, is preferred because it returns a *promise* to give developers access to media devices located either locally or remotely. A promise is a technical term for a proxy value that provides a level of abstraction by promising to supply a value for the media device at a future time to avoid having to immediately return the final value. A promise can be pending, fulfilled or rejected depending on the user's response to grant permission to access their media device. It is set to 'fulfilled' when a user grants permission, to 'rejected' when user denies permission, and 'pending' when no action is performed. Figure 6-15 shows a warning that the old method may cease to work unexpectedly, therefore it is an important aspect to consider when modifying the prototype for further study.

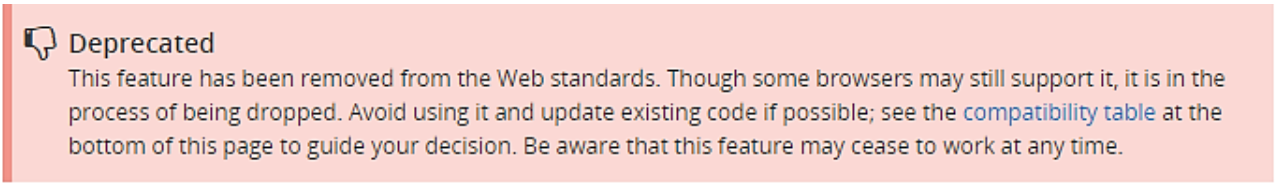

Figure 6-15 - `Navigator.getUserMedia()` deprecated. Source: Mozilla (2017).

6.2.5.2. Secure origins for media (Apache Tomcat, 2017)

As previously mentioned, the strict requirement to secure media through HTTPS origins initially impeded the progress of the work, but was overcome with the purchase of an SSL certificate as a countermeasure. Acquiring a certificate is relatively straightforward and simply follows a step by step process that is well documented in the online community for the different certificate authorities that exist. Apache Tomcat (2017) is an example of an online tutorial available to guide this process. Figure 6-16 depicts the error message displayed in the browser console when attempting to exchange media over insecure channels.

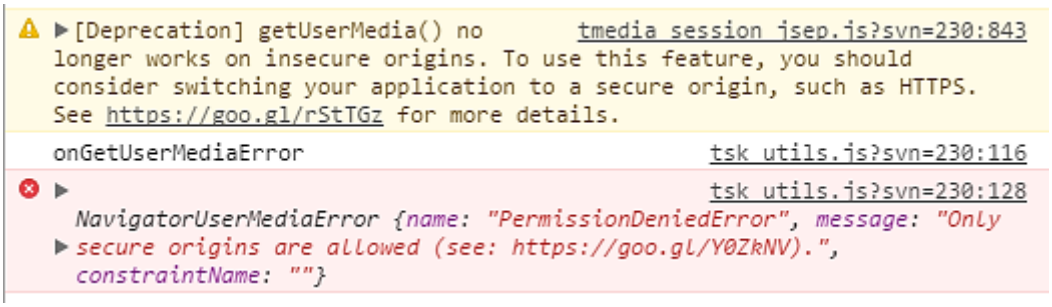

Figure 6-16 - `getUserMedia()` secure origins error on Google Chrome.

6.2.5.3. Session handling scenario

The demonstration of call session handling scenarios posed greater challenges compared to registration scenarios, due to the complex procedures involved when inter-working media, particularly considering the ambiguity of the video codec in the WebRTC landscape. For instance, the noise levels in audio sessions, which used the G.711 audio codec, were quite high and there were delays in the conversation. Video sessions on the other hand proved more challenging. For instance, performing video calling from sipML5 sometimes resulted in abrupt call drops or poor video quality, where at times the call screen would go blank. However, call setup was more seamless when calling sipML5 from IMSDroid. A thorough investigation of the support forums showed that other developers were experiencing similar challenges mostly when integrating WebRTC-based clients with other SIP-based legacy systems (Doubango Telecom, 2016a). Thus, the need for more extensive experiments and learning is necessary to overcome the complexities involved with the correct implementation of a browser RTC “black box” and the codecs, standards, tools and techniques that need to be adopted to support real-time communication.

6.3. Other Tool Considerations

The demonstration of the basic model presents a tool selection that is generally considered as an initial point of reference for experimentation by the Internet community. Otto, Meijer & Skrødal (2016) performed a technology overview of WebRTC interoperability with SIP networks and thus provided reference in addition to Altanai (2014), to consider other potential tools for testing. The purpose of this section is to describe other tools that can be used to realise the model, particularly when implementing the WSF and WMF roles.

An alternative to OpenIMSCore, in the form of Kamailio IMS, is also presented. Kamailio is a powerful tool that undergoes promising technological developments that are both innovative and relevant to the communication needs of the open source VoIP community, incorporating advanced features for supporting TCP, UDP and SCTP transports, secure media communications via TLS, SIMPLE instant messaging and presence, user authentication and authorisation, information storage using databases such as MySQL, PostgreSQL, Oracle and LDAP access, call routing, accounting and many others (Kamailio, 2015).

Kamailio is capable of processing thousands of calls per second and is esteemed as a viable option for SIP routing. As such, an alternative implementation of the model could utilise Kamailio as the WSF, providing support for WebSockets, and for the IMS core network. Kamailio IMS provides a stable architecture for IMS modules since version 4.4 after Fraunhofer FOKUS entrusted the OpenIMSCore development to Core Networks Dynamics. The HSS is still provided by Fraunhofer with the CSCFs configured over Kamailio. DNS configuration for these modules is still required, and the Kamailio configuration file must be modified to enable each entity. Kamailio IMS is generally more complex to set up compared to OpenIMSCore due to additional Diameter configuration files that need to be set up for components with a Diameter interface, namely, the I-CSCF and the S-CSCF.

The WMF role can also be assumed by the FreeSWITCH Media Server which offers full media processing capabilities such as transcoding, call recording, voicemail recording, Interactive Voice Response (IVR) and video conferencing as part of a large carrier-grade telephony framework (West & Boteler, 2017). FreeSWITCH enables transcoding between many audio and video codecs that are available as part of the core or can be compiled and loaded from various modules as per the FreeSWITCH (2012) codecs list. To include the media server in the communication path, the RTP Proxy module is configured in Kamailio to direct media accordingly. The RTP Proxy engine ensures media is relayed appropriately if endpoints are behind NAT and firewalls. Figure 6-17 below depicts the alternative tool selection using jsSIP as the WIC.

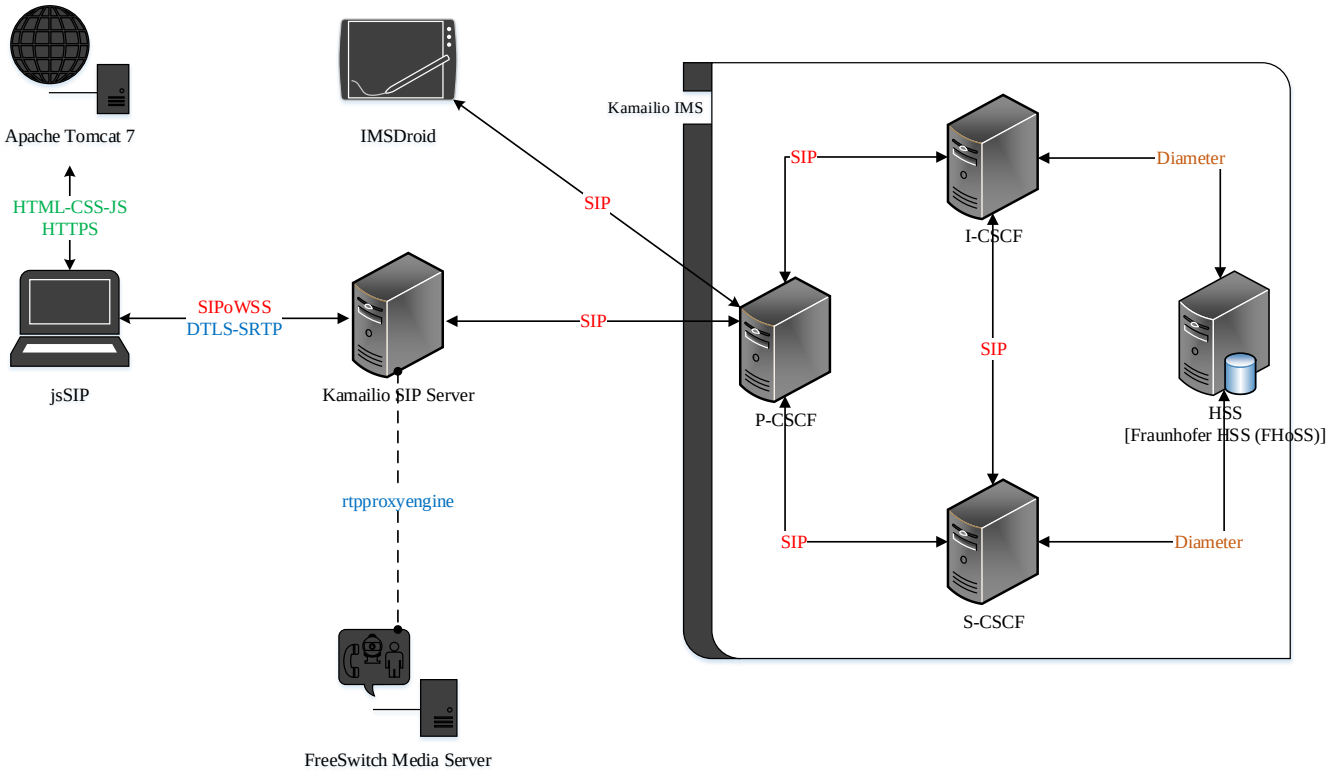

Figure 6-17 - Model implemented using other tools.

The list of open source tools available to enable the integration of WebRTC and IMS is extensive, hence different combinations are possible, although with interoperability challenges. Examples of other frameworks worth considering include the Mobicents Restcomm Communication Platform (Mobicents, 2015). This platform includes a WebRTC AS which can be used to implement the WSF, while its Media WebRTC Server could implement the WMF. In another example, Amirante et al. (2015) describe the Janus WebRTC gateway which is a "barebones" core WebRTC implementation that enables interaction with legacy telco networks over a modular architecture that is capable of supporting signalling alternatives to SIP, basic real-time communication and streaming, video conferencing and server-side techniques to ensure highly scalable and load balanced performance. Kurento is another integration framework that provides signalling and media handling capabilities to provide a powerful modular architecture over which convergent WebRTC and SIP-legacy-based applications are created (Lopez Fernandez et al., 2013). Still more, Ericsson offers OpenWebRTC, a client framework that enables developers to build native mobile applications, and is also based on Gstreamer (Alund, 2015).

This wide tool availability allows the arrangement of different architectures that can address unique scenarios as seen in this chapter. The inability to implement a standalone IMS AGW deviates from the proposed model and is therefore also testament of the efficiency of open source products and the technological advancements they make to enable inter-working between the WebRTC and IMS systems.

6.4. Insights from the Demonstration

The implementation of the model and the review of the open source tools have led to an important observation, one that is summarised in the following point as an emerging requirement to be added to those used in the synthesis of the model:

- **Requirement 6 – the importance of implementing a modular architecture**

As demonstrated by Amirante et al. (2015), it is important to use, as far as possible, tools that are modular to develop an integrated architecture that "allows users to implement a variegated set of advanced services in a scalable fashion" (Amirante et al., 2015). Through the implementation of plugins, the Janus gateway is able to conform to this requirement, as with the tools mentioned within this chapter, particularly Kamailio, whose ability to integrate IMS modules is evidence of the concept put forward to provide a basic model that can be extended through additional supporting functions which can be implemented as modules.

6.5. Conclusion

This chapter demonstrated the use of open source tools that were employed to implement the WebRTC and IMS model in order to demonstrate a successful integration through the demonstration of successful registration and session establishment scenarios. The basic architecture was presented in *Section 6.2* and was realised using common and popular software tools that are used by system integrators. The tools used in this demonstration are sipML5 as the WIC, the webrtc2sip gateway as a combined WSF and WMF, OpenIMSCore as the IMS core network and IMSDroid as the IMS client. The prototype excludes the IMS AGW although it was included as part of the model. The purpose of this change shows the effectiveness of webrtc2sip in performing the necessary media inter-working functions, thus rendering the IMS AGW redundant. In spite of this, the author continues to recognise the importance of the IMS AGW at the conceptual level. *Section 6.2.5* describes the challenges faced when employing these tools, which in some instances extended to the implementation of other tools considered demonstrating the integration that was described in *Section 6.3*. An added feature emergent from the demonstration was identified in *Section 6.4* which expresses the importance of using tools that can be structured and organised into a modular fashion in order to implement a basic model which could be extended via additional functions when executing advanced features. The next and final chapter will then discuss how the implementation realises the thesis objectives and recommendations for future research.

7. Chapter 7 – Conclusion

The purpose of this chapter is to provide concluding remarks that outline the extent to which this work meets the goals defined and the objectives stated for the research. The discussion is structured in such a way as to cross-reference the resultant model and implementation against the original research objectives. The chapter goes on to make recommendations for future work that could be conducted on the network testbed for further experimentation.

7.1. Revisiting the Research Goals

This section summarises the requirements that emerged from the analysis of the IMS service architecture in Chapter 3, the 3GPP investigation of the WebRTC and IMS integration in Chapter 4, as well as the implementation of the proposed model in Chapter 6 which uses open source tools according to the goals and objectives defined for the present research. Consequently, the approach taken to synthesise the research argument is clearly expressed and re-emphasised.

7.1.1. Research Goal 1

To synthesise a WebRTC and IMS integration model that addresses developer needs and requirements.

The discussion of both WebRTC (Chapter 2) and IMS (Chapter 3) systems, particularly the IMS service architecture, provided a coherent argument highlighting common themes that emerged from analysing how service provision occurs in IMS. These themes were organised as requirements which informed a basic integration model that acted as a starting point to discover how telcos are inclined to reuse existing infrastructure to integrate third-party services into their networks. This ability is enabled by the implementation of AS functions which also function as gateways where necessary to connect with ASs in external domains. Furthermore, the heavy reliance on standards and regulations led to the development of standardised interfaces between these functions where internal and external protocols are supported. Thus, WebRTC as a third-party domain of interest, benefits from access to IMS infrastructure as a result of the structures already put in place to enable their integration.

For this purpose, the 3GPP TR 23.701 was extensively analysed in *Section 4.4* to describe how the different architectural solutions propose qualitatively unique candidate integration models. This analysis resulted in the formulation of further requirements that would be added to the espoused integration model. These requirements, and their main elements, summarise the ability to incorporate web-based principles in telco ecosystems, where the use of Operator Web IDs and JSON-based signalling techniques are exemplars. In addition, the requirements also describe an evolutionary measure that telcos can take to extend their existing infrastructure to natively support web techniques. *Section 4.6* covered the 3GPP reference architecture which was used as a guiding framework for the synthesis of the model espoused in this thesis in Chapter 5, which consolidates a practical model for the developer. *Section 5.2.3* presented a discussion of how the functions from the 3GPP reference architecture are conceptualised to develop a “barebones” model using SIP over WebSockets as the main signalling technique and DTLS-SRTP as the main media protocol. This basic view allows one to identify core functions that are required for the integration model to provide core services, following which, any advanced services are decoupled and provided as additional functions required to support the overall architecture.

7.1.2. Research Goal 2

To create an open source testbed that enables testing and experimentation.

The implementation of the model presents a selection of products that could be used to create the network testbed. The use of open source tools was determined to be pragmatic since it would lend unrestricted access to source code, and the array of tools available were effective at experimenting with the standards and protocols required. For instance, the webrtc2sip SIP Proxy module and Kamailio provided WebSocket support, the webrtc2sip RTCWeb Breaker and Media Coder modules, the RTPProxy media engine, FreeSWITCH and others described in *Section 6.3*, provided support for the relevant transcoding and media handling functions. Not only did these tools support the creation of an experimentation platform, they could in theory be arranged in such a way that the model could be realised using different combinations of the tools, with the possibility of extending the testbed further. Thus, the final requirement identified for the research was summarised by the importance of implementing a modular architecture.

7.2. Limitations of the Study

The following sub-sections describe the overall challenges faced when conducting the research which introduced constraints that could influence the quality of the research contribution.

7.2.1. Tool sets

The author recognises that not every available open source tool was tested which could have produced a different implementation.

7.2.2. Training and skills set

The investigation of the available tools presented a steep learning curve. For example, Kamailio requires several interventions in order to run the different modules for WebSockets, IMS, the RTPProxy engine and other fine-tuning for DNS and database access. In addition to analysing files and code, the investigation involved maintaining a high-level view of the overall architecture in order to ensure that other tools could still be able to run and integrate following any modifications. As such, the solution would be challenging for some developers to implement given that it comprises multiple components.

7.2.3. Performance evaluation

The use of testing tools such as SIPp (SIPp, 2014); testRTC (a proprietary WebRTC testing tool) (testRTC, 2017) and Multi-Protocol Test Suite (MTS) (MTS, 2017) could have enhanced the research outcomes by providing a quantitative analysis of the performance of the integration model, thus giving a better perspective of the qualitative accomplishments of this research. Cruz & Barraca (2015) for instance, evaluate the performance of a WebRTC and IMS system based on Solution 5 of TR 23.701 by measuring the call throughput and mouth-to-ear delay using MTS. They suggest that call delays experienced over an integrated architecture are similar to those experienced with mobile network calls. Furthermore, these testing tools could enable the creation of data sets that could be used to measure different tools' capabilities when trying a variety of communication scenarios. Adeyeye et al. (2013) is another example of a study that could have been conducted where signalling overheads of different protocols are compared. Even though this facility would have been beneficial, it was never a goal of the work hence the focus on synthesising the design.

7.3. Recommendations for Future Work

The current implementation provides a basic model with the potential for the inclusion of support for advanced features illustrated by the WIC architecture in Figure 5-5, thus creating numerous

opportunities for further research to be conducted over the network testbed. As such, opportunities stem from deploying these advanced features whereas other opportunities from addressing the limitations of the study described in the previous section. Therefore, some examples of further research that could be conducted are summarised in the following subsection.

7.3.1. Identity Management

It would be desirable to extend the testbed through added identity management functions such as implementing operator web identities using SIP or another mechanism, or modelling identity management provided by the operator or a web-based IdP such as Google within a mixed WebRTC-IMS context. This use case could involve testing SIP OAuth2.0 on a WIC as proposed by Shekh-Yusef & Pascual (2014). The SIM authentication scheme proposed by Solution 7 in *Section 4.4.7* is another instance that could be realised for this use case that also supports the investigation of the WWPF as a unique function suggested for this architecture.

7.3.2. Signalling Alternatives

The issue of signalling alternatives to SIP could be investigated. XMPP could be investigated in addition to using transport channels that are different from WebSockets such as XHR or even WebRTC Data Channels. The study conducted by Adeyeye et al. (2013) is evidence of the feasibility of alternative signalling protocols and transport alternatives that can be implemented. The use of WebRTC Data Channels is also another mechanism that offers diverse usage in terms of transporting JSON-based or proprietary signalling messages along the control plane, and can therefore be used within applications that do not need a centralised server to setup Data Channels, for instance during live gaming and P2P file sharing.

7.3.3. Integration with other Domains

The integration of WebRTC services with RCS could be investigated where the session handling scenario described for a WIC implementing an RCS messaging service could be modelled. Another example of is PSTN interworking, where a telecom server such as the Mobicents AS could be used as the integration tool to enable services such as IVR, voice mail and other call handling capabilities.

7.4. Statement of Contributions

The research has contributed:

1. A WebRTC and IMS model that meets developers' needs for experimentation.

Developers can benefit from the practicality and ease of integration of the proposed model which has been designed to leverage existing IMS infrastructure while also providing a forward-thinking view by enabling evolution through additional functions. The thesis also acts as a guiding framework for developers looking to understand the implications of integrating WebRTC and IMS by using a model whose requirements conform to standards prescribed by standardisation bodies.

2. A synopsis of open source tools available to support the integration of WebRTC with IMS.

The description of the implementation process gives a synopsis of tool availability and support, mainly for SIP-based WebRTC systems but also provides easy access to implementing other protocols by supporting additional functions where necessary, thus improving efficacy when making development decisions.

List of References

3GPP. 2008a. *Open Service Access (OSA) Application Programming Interface (API); Part 10: Connectivity Manager Service Capability Feature (SCF)*. Retrieved (http://www.3gpp.org/ftp/Specs/html-info/29198-10.htm).

3GPP. 2008b. *Policy and Charging Control (PCC) over S9 Reference Point*. Retrieved (http://www.3gpp.org/ftp/Specs/html-info/29215.htm).

3GPP. 2013. *Study on Web Real Time Communication (WebRTC) Access to IP Multimedia Subsystem (IMS); Stage 2 (Release 12)*. Retrieved (http://www.3gpp.org/DynaReport/23701.htm).

3GPP. 2015. "IP Multimedia Subsystem (IMS); Stage 2 (Release 14)." (Stage 2):1–311. Retrieved (https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=821).

3GPP. 2017. "IP Multimedia Subsystem." *3GPP - A Global Initiative* 1. Retrieved July 19, 2017 (http://www.3gpp.org/technologies/keywords-acronyms/109-ims).

Adeyeye, Michael, Ishmeal Makitla, and Thomas Fogwill. 2013. "Determining the Signalling Overhead of Two Common WebRTC Methods: JSON via XMLHttpRequest and SIP over WebSocket." in *IEEE AFRICON Conference*.

Alexandru, Carol. 2014. *Impact of WebRTC (P2P in the Browser)*. Zurich, Switzerland.

Altanai. 2014. *WebRTC Integrator's Guide*. First. edited by A. Arrichiello, P. Boemio, A. R. Portabales, and A. Sergiienko. Birmingham: Packt Publishing Ltd. Retrieved (http://www.it-ebooks.info/book/4643/).

Alund, Stefan. 2015. "OpenWebRTC - What's Happening 2015?" *Ericsson Research Blog* 1. Retrieved April 28, 2015 (http://www.ericsson.com/research-blog/context-aware-communication/openwebrtc-whats-happening-2015/).

Alvestrand, Harald and Adrian Grange. 2013. *VP8 as RTCWEB Mandatory to Implement*. Retrieved (http://www.ietf.org/internet-drafts/draft-alvestrand-rtcweb-vp8-02.txt).

Amirante, Alessandro, Tobia Castaldi, Lorenzo Miniero, and Simon Romano. 2013. "On the Seamless Interaction between webRTC Browsers and SIP-Based Conferencing Systems." *IEEE Communications Magazine* 51(4):42–47.

Amirante, Alessandro, Tobia Castaldi, Lorenzo Miniero, and Simon Pietro Romano. 2015. "Performance Analysis of the Janus WebRTC Gateway." P. 4:1--4:7 in *Proceedings of the 1st Workshop on All-Web Real-Time Systems*, *AWeS '15*. New York, NY, USA: ACM. Retrieved (http://doi.acm.org/10.1145/2749215.2749223).

Apache Tomcat. 2017. *SSL/TLS Configuration HOW-TO*. Retrieved (https://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html).

Asterisk. 2017. "Asterisk." *asterisk.org* 1. Retrieved June 1, 2017 (http://www.asterisk.org/).

Bach, Tilmann, Michael Maruschke, Jens Zimmermann, H. Kay, and Matthias Baumgart. 2014. "Combination of IMS-Based IPTV Services with WebRTC." Pp. 140–45 in *The Ninth International Multi-Conference on Computing in the Global Information Technology, ICCGI 2014*.

Bakore, Amit. 2003. *Professional Apache Tomcat*. Wrox. Retrieved May 16, 2017 (https://books.google.co.za/books?id=6lXRnoVEOQoC&printsec=frontcover&source=gbs_atb#v=onepage&q&f=false).

Bankoski, J., Paul Wilkins, and Yaowu Xu. 2011. Technical overview of VP8, an open source video codec for the web. *Multimedia and Expo (ICME), 2011 IEEE International Conference on.* Barcelona: IEEE.

Beltran, Victoria, Emmanuel Bertin, and Noël Crespi. 2014. "User Identity for WebRTC Services: A Matter of Trust." *IEEE Internet Computing* 18(6):18–25.

Benali, O., K. El-Khazen, D. Garrec, M. Guiraudou, and G. Martinez. 2004. "A Framework for an Evolutionary Path toward 4G by Means of Cooperation of Networks." *IEEE Communications Magazine* 42(5):82–89.

Bertin, Emmanuel, Noel Crespi, and Michel L'Hostis. 2011. "A Few Myths about Telco and OTT Models." Pp. 6–10 in *2011 15th International Conference on Intelligence in Next Generation Networks, ICIN 2011*.

Bertin, Emmanuel, Sébastien Cubaud, Stéphane Tuffin, Noël Crespi, and Victoria Beltran. 2013. "WebRTC, the Day after: What's next for Conversational Services?" *2013 17th International Conference on Intelligence in Next Generation Networks, ICIN 2013* (January):46–52.

Bertin, Emmanuel, Imen Ben Yahia, and Noel Crespi. 2007. "Modeling IMS Services." *Journal of Mobile Multimedia* 3(2):150–67. Retrieved (http://www.it-sudparis.eu/dpt/rs2m/ncpub/2006/Journal of Mobile Multimedia/JMM final.pdf).

Bertrand, Gilles. 2007. "The IP Multimedia Subsystem in Next Generation Networks." *Network, Multimedia and Security department (RSM)- …* 7(March):1–9. Retrieved (http://www1.coe.neu.edu/~eeichen/spring_2013/class_notes/j_march_14/IMS_an_overview.pdf).

Black, U. D. 2001. *Internet Telephony: Call Processing Protocols*. 1st ed. Prentice Hall PTR. Retrieved (https://books.google.co.za/books?id=JPhSAAAAMAAJ).

Brouquet, Daniel. 2008. *Pervasive Networks and Connectivity Seminar Series on Special Topics in Networking: IP Multimedia Subsystem, Spring 2008*.

Camarillo, Gonzalo and Miguel-Angel Garcia-Martin. 2007. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds.* Third. West Sussex: John Wiley & Sons.

Cardoza, Christina. 2015. "WebRTC: The Road to Standardization - SD Times." *Software Development Times* 3. Retrieved October 25, 2016 (http://sdtimes.com/webrtc-road-standardization/).

Casner, Steve. 2016. "Real-Time Transport Protocol (RTP) Parameters." *Internet Assigned Numbers Authority (IANA)* 1. Retrieved October 7, 2016 (http://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml).

Chrome Help. 2015. "Blocked Plugins." Retrieved May 8, 2015 (https://support.google.com/chrome/answer/1247383?hl=en).

Core Network Dynamics. 2017. "About | Germany | Core Network Dynamics." *Core Network Dynamics* 1. Retrieved August 16, 2017 (https://www.corenetdynamics.com/about).

Crockford, D. 2006. *The Application/json Media Type for JavaScript Object Notation (JSON)*. RFC Editor. Retrieved (http://www.rfc-editor.org/rfc/rfc4627.txt).

Cruz, B. S. and J. P. Barraca. 2015. “IMS Centric Communication Supporting WebRTC Endpoints.” Pp. 732–37 in *2015 IEEE Symposium on Computers and Communication (ISCC)*.

Davies, Marcin, Joachim Zeiss, and Rene Gabner. 2012. “Evaluating Two Approaches for Browser-Based Real-Time Multimedia Communication.” P. 109 in *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*. Retrieved (http://dl.acm.org/citation.cfm?doid=2428955.2428982).

Doubango Telecom. 2016a. “Doubango Telecom webrtc2sip.” *GitHub* 1. Retrieved (https://github.com/DoubangoTelecom/webrtc2sip/issues).

Doubango Telecom. 2016b. “Smart SIP and Media Gateway to Connect WebRTC Endpoints.” Retrieved (https://www.doubango.org/webrtc2sip/).

Doubango Telecom. 2018. "Doubango Telecom." Retrieved (https://www.doubango.org/).

Dutton, Sam. 2015. “Chrome 47 WebRTC: Media Recording, Secure Origins & Proxy Handling.” *Google Developers*. Retrieved (https://developers.google.com/web/updates/2015/10/chrome-47-webrtc?hl=en).

Eisenmann, Thomas R., Geoffrey Parker, and Marshall Van Alstyne. 2008. *Opening Platforms: How, When and Why?* Cheltenham,. Retrieved July 26, 2017 (http://www.hbs.edu/faculty/Publication Files/09-030.pdf).

El Alaoui, Sara et al. 2012. “Towards Future 4G Mobile Networks: A Real-World IMS Testbed.” *International Journal of Next-Generation Networks (IJNGN)* 4(3). Retrieved May 15, 2017 (http://airccse.org/journal/ijngn/papers/4312ijngn03.pdf).

Eriksson, GP and S. Hakansson. 2012. “WebRTC: Enhancing the Web with Real-Time Communication Capabilities.” *Ericson Rev* 5–9. Retrieved June 14, 2015 (http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:WebRTC+:+enhancing+the+web+with+real-time+communication+capabilities#0).

ETSI. 2007. “ETSI - Common IMS to Be Centred in the 3GPP Services Specification Group.” *ETSI News* 1. Retrieved July 25, 2017 (http://www.etsi.org/news-events/news/202-news-release-18th-june-2007).

Fette, Ian and Alexey Melnikov. 2011. “The WebSocket Protocol.” Retrieved (https://tools.ietf.org/html/rfc6455).

FreeSwitch. 2012. “Codecs - FreeSWITCH Wiki.” *FreeSwitch Wiki* 1. Retrieved June 7, 2017 (https://wiki.freeswitch.org/wiki/Codecs).

Friese, I. et al. 2010. “Bridging IMS and Internet Identity.” Pp. 1–6 in *Intelligence in Next Generation Networks (ICIN), 2010 14th International Conference on*.

Ghadialy, Zahid. 2004. “CAMEL: An Introduction.” *3G4G.org* 1. Retrieved December 2, 2016 (http://www.3g4g.co.uk/Tutorial/ZG/zg_camel.html).

Google Groups. 2015. *Google groups forum - discuss doubango*. Retrieved (https://groups.google.com/forum/#!topic/doubango/-6XKVB_Y1kY).

GNU. 2007. “GNU General Public License.” *GNU Operating System* 1. Retrieved August 16, 2017 (https://www.gnu.org/licenses/gpl.html).

Haas, Hugo and Allen Brown. 2004. “W3C. Web Services Glossary. Definitions. Web Service.” *W3C* 3(February):1–17. Retrieved December 9, 2016 (https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice).

Higa, D. 2008. “Walled Gardens versus the Wild West.” *Computer* 41(10):102–5.

Hirsch, Frederick and Andrew Braun. 2010. “Ubiquitous Web Applications Activity Statement.” *Ubiquitous Web Domain* (September):2010–11. Retrieved (https://www.w3.org/2007/uwa/Activity.html).

Holmberg, C., S. Hakansson, and G. Eriksson. 2013. “Web Real-Time Communication Use-Cases and Requirements.” *draft-ietf-rtcweb-usecases-and-requirements-11*. Retrieved (https://tools.ietf.org/html/rfc7478).

Howes, Tim, Mark Smith, and Gordon S. Good. 2003. “Introduction to Directory Services and LDAP.” P. 899 in *Understanding and Deploying LDAP Directory Services*. Boston: Addison-Wesley.

IETF. 2014. “Overview: Real Time Protocols for Browser-Based Applications.” 3–22. Retrieved May 18, 2015 (http://datatracker.ietf.org/doc/draft-ietf-rtcweb-overview/?include_text=1).

ITU. 2016. *Measuring the Information Society Report*. Geneva, Switzerland. Retrieved July 23, 2017 (http://www.itu.int/en/ITU-D/Statistics/Documents/publications/misr2016/MISR2016-w4.pdf).

ITU-T. 1998. *T.140: Protocol for Multimedia Application Text Conversation*. Retrieved (https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-T.140-199802-I!!PDF-E&type=items).

Janczukowicz, Ewa, Ahmed Bouabdallah, and Jean-marie Bonnin. 2015. “Specialized Network Services for WebRTC.” P. 6 in *AWeS ’15 Proceedings of the 1st Workshop on All-Web Real-Time System*. New York. Retrieved (http://dl.acm.org/ft_gateway.cfm?id=2749218&ftid=1564576&dwn=1&CFID=713346403&CFTOKEN=23309977).

Jennings, C., Peterson, J., & Watson, M. (2002). *Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks.* RFC Editor. RFC Editor.

Jesup, Randell, Salvatore Loreto, and Michael Tuexen. 2015a. *WebRTC Data Channel Establishment Protocol*. Retrieved (http://www.ietf.org/internet-drafts/draft-ietf-rtcweb-data-protocol-09.txt).

Jesup, Randell, Salvatore Loreto, and Michael Tuexen. 2015b. *WebRTC Data Channels*. Retrieved (http://www.ietf.org/internet-drafts/draft-ietf-rtcweb-data-channel-13.txt).

Jitsi. 2011. “A SIP to Jingle (XMPP) Gateway in Kamailio (OpenSER) | Jitsi.” *Jitsi.org* 1. Retrieved November 16, 2016 (https://jitsi.org/GSOC2011/KamailioJingle).

Jobs, Steve. 2010. “Thoughts on Flash.” *Apple*. Retrieved May 6, 2015

(http://www.apple.com/hotnews/thoughts-on-flash/).

Johnston, Alan B. and Daniel C. Burnett. 2013. “WebRTC: The Web Way to Communicate.” 10. Retrieved (http://webrtcbook.com/presentations/WebRTCIEEE04-02-13.pdf).

Johnston, Alan, John Yoakum, and Kundan Singh. 2013. “Taking on WebRTC in an Enterprise.” *IEEE Communications Magazine* 51(4):48–54.

Kamailio. 2015. “Welcome to Kamailio - the Open Source SIP Server.” 1. Retrieved June 12, 2015 (http://www.kamailio.org/w/).

Kaplan, Hadriel. 2015. *Should We Support SDES in WebRTC?* Retrieved June 6, 2017 (https://www.ietf.org/proceedings/84/slides/slides-84-rtcweb-15.pdf).

Khandelwal, Rakesh. 2007. “The Importance of Standard IMS Architecture.” *Architecture* 1–7. Retrieved (http://blog.pucp.edu.pe/blog/wp-content/uploads/sites/100/2007/09/Importance-of-IMS.pdf).

Khlifi, Hechmi and Jean Charles Grégoire. 2008. “IMS Application Servers: Roles, Requirements, and Implementation Technologies.” *IEEE Internet Computing* 12(3):40–51.

Kurose, James F. and Keith W. Ross. 2012. *Computer Networking: A Top-Down Approach (6th Edition)*. 6th ed. Pearson.

Levent-Levi, Tsahi. 2014. “The Real Codec Battle Is VP9 vs H.265 - And VP9 Is Winning - Post - No Jitter.” *NoJitter* 1. Retrieved November 16, 2016 (http://www.nojitter.com/post/240168581/the-real-codec-battle-is-vp9-vs-h265--and-vp9-is-winning).

Lopez Fernandez, L. and Paris Diaz, M. and Benitez Mejias, R. and Lopez, F.J. and Santos, J. A. 2013. “Kurento: A Media Server Technology for Convergent WWW/mobile Real-Time Multimedia Communications Supporting WebRTC.” Pp. 1–6 in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. Madrid: IEEE Xplore Digital Library. Retrieved (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6583507&isnumber=6583357).

Loreto, Salvatore; and Simon Romano. 2014. *Real-Time Communication with WebRTC Peer-to-Peer in the Browser*. 1st ed. Safari Books Online: O’Reilly Media.

Ludwig, Scott, Joe Beda, Peter Saint-Andre, Robert McQueen, Sean Egan, and Joe Hildebrand. 2016. XEP-0166: *Jingle*. XMPP Standards Foundation. Retrieved (https://xmpp.org/extensions/xep-0166.html).

Lynch, Lucy. 2011. “Inside the Identity Management Game.” *IEEE Internet Computing* 15(5):78–82.

Maes, Stéphane H. 2010. “Next Generation Telco Service Providers : Telco 2 . 0 and Beyond.” Huawei CTO Whitepaper. Retrieved (http://www.stephanemaes.com/ESSEM/Download/2010/Huawei_CTO_paper_sm_7_5_10.pdf)

Magedanz, Thomas, Niklas Blum, and Simon Dutkowski. 2007. “Evolution of SOA Concepts in Telecommunications.” *Computer* 40(11):46–50.

Mahy, R., P. Matthews, and J. Rosenberg. 2010. *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)*. RFC Editor. Retrieved (http://www.rfc-editor.org/rfc/rfc5766.txt).

McGrew, D. 2010. "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-Time Transport Protocol (SRTP)(RFC 5764), IETF." Retrieved (https://tools.ietf.org/html/rfc5764).

Microsoft Developers. 2016. "Dev Guide: Object RTC API - Microsoft Edge Development." *Microsoft Developers* 1. Retrieved (https://developer.microsoft.com/en-us/microsoft-edge/platform/documentation/dev-guide/realtime-communication/object-rtc-api/).

Minerva, Roberto and Steve Bell. 2010. "Boundary Blurring between Telecom and the Internet." *EMEA 2010* 8–11.

Miniero, Lorenzo, Alessandro Amirante, Tobia Castaldi, and Simon Romano. 2008. *A Binary Floor Control Protocol (BFCP) Control Package for the Session Initiation Protocol (SIP)*. Retrieved (http://www.ietf.org/internet-drafts/draft-miniero-bfcp-control-package-00.txt).

Mobicents. 2015. "The Mobicents Communication Platform." Retrieved May 18, 2015 (http://www.mobicents.org/).

Moerdijk, A. J. and Lucas Klostermann. 2003. "Opening the Networks with Parlay/OSA: Standards and Aspects behind the APIs." *IEEE network* 17(3):58–64.

Mozilla. 2015. "Plugins." *Mozilla Developer Network*. Retrieved May 5, 2015 (https://developer.mozilla.org/en-US/Add-ons/Plugins).

Mozilla. 2016. "Performance.now() - Web APIs | MDN." *Mozilla Foundation* 1. Retrieved (https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection/onicecandidate).

Mozilla. 2017. "Navigator.getUserMedia() - Web APIs | MDN." *Mozilla Developer Network* 1. Retrieved May 22, 2017 (https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia#Browser_compatibility).

MTS. 2017. "Multi-Protocol Test Suite : The Solution to Integrate, Test and Optimize Your IP Telecom System." *Ericsson MTS* 1. Retrieved June 7, 2017 (http://mts.arm-tool.com/).

Mulligan, C. E. A. 2009. "Open API Standardization for the NGN Platform." *IEEE Communications Magazine* 47(5):108–13.

Muranyi, J. and I. Kotuliak. 2013. "Identity Management in WebRTC Domains." Pp. 289–93 in *Emerging eLearning Technologies and Applications (ICETA), 2013 IEEE 11th International Conference on*. Retrieved (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6674445%5Cnpapers3://publication/doi/10.1109/ICETA.2013.6674445).

Muswera, Wt and Alfredo Terzoli. 2010. "Development of an IMS Compliant, Cross Platform Client Using the JAIN SIP Applet Phone." in *Development of an IMS Compliant, Cross Platform Client Using the JAIN SIP Applet Phone*. Retrieved (http://www.satnac.org.za/proceedings/2010/papers/poster/Muswera 490.pdf).

Narbutt, Miroslaw and Mark Davis. 2005. "An Assessment of the Audio Codec Performance in Voice

over WLAN (VoWLAN) Systems." Retrieved November 12, 2016 (http://arrow.dit.ie/commcon).

NetCraft. 2017. "February 2017 Web Server Survey | Netcraft." *Netcraft News* 1. Retrieved May 16, 2017 (https://news.netcraft.com/archives/2017/02/27/february-2017-web-server-survey.html).

O'Connell, John. 2007. "Service Delivery within an IMS Environment." *IEEE Vehicular Technology Magazine* 2(1):12–19.

Olanoff, Drew. 2015. "Google Acquires Jibe Mobile To Help Adopt New Standard For Carrier Messaging | TechCrunch." *Tech Crunch* 1. Retrieved February 26, 2017 (https://techcrunch.com/2015/09/30/google-acquires-jibe-mobile-to-help-adopt-new-standard-for-carrier-messaging/).

Open Mobile Alliance. 2005. "Utilization of IMS Capabilities Requirements." 1–15. Retrieved October 26, 2016 (http://technical.openmobilealliance.org/Technical/Release_Program/docs/IMS/V1_0-20050809-A/OMA-RD-IMSinOMA-V1_0-20050809-A.pdf).

openimscore.org. 2015. "OpenIMS – The Open Source IMS Core Project." *Core Network Dynamics* 1. Retrieved May 20, 2017 (http://www.openimscore.org/).

Otto, Stefan, Jan Meijer, and Simon Skrødal. 2016. *SA8T2 Internal Deliverable. Technology Scout: WebRTC2SIP Gateway*.

Pascual, Victor. 2014. "The IMS Approach to WebRTC - webrtcHacks." *webrtcH4cKS* 1. Retrieved February 9, 2017 (https://webrtchacks.com/ims-approach-webrtc/).

Pascual, Victor Ávila. 2013. "WebRTC MUST Implement DTLS-SRTP But… MUST NOT Implement SDES?" *webrtcH4cKS* 1. Retrieved June 6, 2017 (https://webrtchacks.com/webrtc-must-implement-dtls-srtp-but-must-not-implement-sdes/).

Pimentel, Victoria and Bradford G. Nickerson. 2012. "Communicating and Displaying Real-Time Data with WebSocket." *IEEE Internet Computing* 16(4):45–53.

Prasad, J.Kalyan and B. Anil Kumar. 2011. "Analysis of SIP and Realization of Advanced IP-PBX Features." Pp. 218–22 in *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, vol. 6.

Proust, S. et al. 2015. *Additional WebRTC Audio Codecs for Interoperability*. *RFC Editor*. Retrieved (https://tools.ietf.org/html/rfc7875).

Raivio, Yrjo, and Sakari Luukkainen. 2011. "Mobile Networks as a Two-Sided Platform - Case Open Telco." *Journal of Theoretical and Applied Electronic Commerce Research* 6(2):77–89.

Ravindran, Parthasarathi, Uwe Rauschenbach, and Elangovan Manickam. 2013. *Offer & Answer Interworking between JSEP & SIP*. Retrieved (http://www.ietf.org/internet-drafts/draft-partha-rtcweb-jsep-sip-01.txt).

Raymond, Robin. 2012. "Open Peer - A Proposed Peer-to-Peer Signaling Protocol for WebRTC." *Hookflash* 1–11. Retrieved April 27, 2015 (https://www.scribd.com/doc/114565509/Open-Peer-for-WebRTC-Whitepaper).

Reichl, Peter, Sandford Bessler, Joachim Fabini, Rudolf Pailer, and Joachim Zeiss. 2006. "Implementing a Native IMS Location Service Enabler over a Prototypical IMS Core Network Testbed." Pp. 2–9 in *CEC/EEE 2006 Joint Conferences*, vol. 2006.

Rescorla, Eric. 2015a. *Security Considerations for WebRTC*. Retrieved (http://www.ietf.org/internet-drafts/draft-ietf-rtcweb-security-08.txt).

Rescorla, Eric. 2015b. *WebRTC Security Architecture*. Retrieved (http://www.ietf.org/internet-drafts/draft-ietf-rtcweb-security-arch-11.txt).

Richardson, Texas. 2014. "Mavenir Selected by MTS for Advanced Multimedia Services Based on RCS | ITWeb." *IT Web Telecoms* 1. Retrieved February 26, 2017 (http://www.itweb.co.za/index.php?option=com_content&view=article&id=71400:Mavenir-selected-by-MTS-for-advanced-multimedia-services-based-on-RCS&catid=260).

Roach, A. B. and Mozilla. 2015. "WebRTC Video Processing and Codec Requirements." Retrieved (https://tools.ietf.org/html/draft-ietf-rtcweb-video-06).

Romain, Carbou. 2013. "Some WebRTC Opportunities for RCS: And Some Inner Challenges to Overcome." *2013 17th International Conference on Intelligence in Next Generation Networks, ICIN 2013* 31–38.

Rosenberg, J. 2010. *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*. RFC Editor. Retrieved (http://www.rfc-editor.org/rfc/rfc5245.txt).

Rosenberg, J. and H. Schulzrinne. 2002. *An Offer/Answer Model with Session Description Protocol (SDP)*. RFC Editor. Retrieved (http://www.rfc-editor.org/rfc/rfc3264.txt).

Rosenberg, Jonathan, Matthew Kaufman, Magnus Hiie, and Francois Audet. 2011. *An Architectural Framework for Browser Based Real-Time Communications (RTC)*. Retrieved (http://www.ietf.org/internet-drafts/draft-rosenberg-rtcweb-framework-00.txt).

Sansay. 2013. "Integrating WebRTC with Existing VoIP Networks." 1–7. Retrieved (http://www.sansay.com/sandbox/wp-content/uploads/2013/06/Integrating_WP_062313_FINAL_3.pdf).

Schuh, Justin. 2013. "Saying Goodbye to Our Old Friend NPAPI." *Chromium Blog*. Retrieved May 5, 2015 (http://blog.chromium.org/2013/09/saying-goodbye-to-our-old-friend-npapi.html).

Schulzrinne, H., S. Casner, R. Frederick, and V. Jacobson. 2003. *RTP: A Transport Protocol for Real-Time Applications*. RFC Editor. Retrieved (http://www.rfc-editor.org/rfc/rfc3550.txt).

Sege, Pavel, Peter Palúch, Jozef Papán, and Milan Kubina. 2014. "The Integration of WebRTC and SIP : Way of Enhancing Real-Time , Interactive Multimedia Communication." Pp. 437–42 in *Proceedings of the 12th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. IEEE Xplore Digital Library.

Segec, Pavel and Tatiana Kovacikova. 2012. "Implementation Of IMS Testbeds Using Open Source Platforms." 8. Retrieved May 15, 2017 (https://www.academia.edu/11040281/IMPLEMENTATION_OF_IMS_TESTBEDS_USING_OPEN_SOURCE_PLATFORMS).

Shekh-Yusef, Rifaat and Victor Pascual. 2014. *The Session Initiation Protocol (SIP) OAuth*. Retrieved (http://www.ietf.org/internet-drafts/draft-yusef-sipcore-sip-oauth-00.txt).

Shores, Redwood, Castro Valley, Honggang Frank Zhu, and Karthic Loganathan. 2014. "System And Method For Extending IP Multimedia Subsystem To HTML5 Environments." 1(19):14.

sipML5. 2017. "Doubango Telecom / sipML5." 1. Retrieved (https://github.com/DoubangoTelecom/sipml5).

SIPp. 2014. "Welcome to SIPp." *SIPp Sourceforge* 1. Retrieved June 7, 2017 (http://sipp.sourceforge.net/).

Skvorc, D., M. Horvat, and S. Srbljic. 2014. "Performance Evaluation of Websocket Protocol for Implementation of Full-Duplex Web Streams." Pp. 1003–8 in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014 - Proceedings*.

Spiers, Richard and Neco Ventura. 2010. "A Converged IMS Client for the IP Multimedia Subsystem." in *Rondebosch, South Africa: University of Cape …*. Retrieved (http://www.satnac.org.za/proceedings/2010/papers/software/Spiers FP 384.pdf).

Sredojev, Branislav, Dragan Samardzija, and Dragan Posarac. 2015. "WebRTC Technology Overview and Signaling Solution Design and Implementation." Pp. 1006–9 in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings*.

Stackoverflow. 2016. *JavaScript GetUserMedia using Chrome with localhost without HTTPS*. Retrieved (https://stackoverflow.com/questions/40144036/javascript-getusermedia-using-chrome-with-localhost-without-https).

StatCounter.com. (2016). *Top 5 Desktop, Tablet & Console Browsers from Oct 2015 to Oct 2016 | StatCounter Global Stats*. Retrieved from StatCounter: Global Stats: http://gs.statcounter.com/

STL Partners. 2015. *The Open Source Telco: Taking Control of Destiny - STL Partners / Telco 2.0 Research*. Retrieved July 26, 2017 (https://www.telco2research.com/articles/EB_the-open-source-telco).

Talky. 2017. "Browser Support Scorecard - Is WebRTC Ready Yet?" *Talky* 1. Retrieved May 18, 2017 (http://iswebrtcreadyyet.com/).

Taylor, R. and J. Ing. 2013. "WebRTC Overview - 3GPP." Presentation: Public Safety Canada.Retrieved (ftp://www.3gpp.org/TSG_SA/WG3_Security/TSGS3_LI/2013_51_Burlington/SA3LI13_136r1.ppt).

testRTC. 2017. "Homepage testRTC." *testRTC* 1. Retrieved June 7, 2017 (https://testrtc.com/).

Toutain, François, Emmanuel Le Huérou, and Eric Beaufils. 2015. "On Webco Interoperability." Pp. 1–6 in *Proceedings of the 1st Workshop on All-Web Real-Time Systems - AWeS '15*. Retrieved (http://dl.acm.org/citation.cfm?doid=2749215.2749219).

TSGC. 2015. *TS 124 371 - V12.0.0 - Universal Mobile Telecommunications System (UMTS); LTE; Web Real-Time Communications (WebRTC) Client Access to the IP Multimedia (IM) Core Network*

(CN) Subsystem; Protocol Specification (3GPP TS 24.371 Version 12.0.0 Release 12). Sophia Antipolis Cedex. Retrieved February 10, 2017 (http://www.etsi.org).

Tsietsi, M., S. Honye, and H. Thinyane. 2015. "Modelling the Exposure of Services within next Generation Telecommunication Networks." Pp. 1–11 in *IST-Africa Conference, 2015*, edited by P. Cunningham and M. Cunningham.

Uberti, Justin, Cullen Jennings, and Eric Rescorla. 2015. *Javascript Session Establishment Protocol*. Retrieved (http://www.ietf.org/internet-drafts/draft-ietf-rtcweb-jsep-16.txt).

Ubiquity. 2005. "A Concise Guide To The Major Internet Bodies." *Association for Computing Machinery* 1. Retrieved (http://ubiquity.acm.org/article.cfm?id=1071915).

VoipSwitch. 2014. "What Makes a Native OTT or RCS Mobile Client WebRTC Compatible? -." *VoipSwitch* 1. Retrieved May 4, 2017 (http://www.voipswitch.com/what-makes-a-native-ott-or-rcs-mobile-client-webrtc-compatible/).

Vukotic, Aleksa and James Goodwill. 2011. "Integrating Apache Web Server." Pp. 185–97 in *Apache Tomcat 7*. Berkeley, CA: Apress. Retrieved May 16, 2017 (http://link.springer.com/10.1007/978-1-4302-3724-2_10).

W3C. 2009. "Device and Sensors Working Group - W3C." *W3C* 1. Retrieved (https://www.w3.org/2009/dap/#mediacapture).

W3C. 2015. "WebRTC 1.0: Real-Time Communication Between Browsers." *W3C Working Draft*. Retrieved May 18, 2015 (http://www.w3.org/TR/webrtc/).

WebKit. 2017. "WebKit Feature Status | WebRTC Specification." *WebKit* 1. Retrieved November 2, 2017 (https://webkit.org/status/#specification-webrtc).

West, Brian and John Boteler. 2017. "FreeSWITCH Explained - FreeSWITCH - Confluence." *FreeSwitch* 1. Retrieved June 7, 2017 (https://freeswitch.org/confluence/display/FREESWITCH/FreeSWITCH+Explained).

York, Dan. 2013. "WebRTC: Moving Real-Time Communication into the Web Browser." *The IETF Journal* 9(1):11–12. Retrieved (http://www.internetsociety.org/sites/default/files/IETF 86_July 11b-1.pdf).