EVOLVING THE TELECOM STACK AND HOW WEBRTC PLAYS A ROLE

Chris Wendt, Principal Architect, IP Communications and Services
Comcast

Abstract

This paper provides an overview and rationale for a vision of the future evolution of telecommunications services and how real-time communications can be provided as a more flexible web oriented service. WebRTC, a web browser focused framework for real-time communications, provides standardized client-side "hooks" which can be adapted for mobile, set-top box and other popular embedded clients. The larger question is how these client technologies can be used as part of a modern, web-friendly service framework. We at Comcast Labs are proposing a framework and perspective beyond traditional telephony services, defining how real-time communications can be customized and integrated for the variety of service contexts and devices we use in our personal and professional lives. Examples include: enhanced customer service, enhanced home monitoring and device control, interactive collaboration or shared experiences. This paper will propose and discuss a framework and architecture and provide the motivation why WebRTC plays an important role, and discuss why past attempts may not have been as successful, and why based on technology and network evolution, and device capabilities, this time might be different.

INTRODUCTION

The way humans communicate in just the past five or so years has evolved at a lightning pace. With the now nearly ubiquitous availability of bandwidth and communications over wired and wireless networks and the exponential improvement in CPU and GPU capabilities of devices, the world has changed for how the human/computer interface has evolved, how we interact with our devices, how we buy and consume content and applications. It's almost unthinkable how quickly humans have adopted these new paradigms without looking back. For these reasons and others, our communications habits have been similarly transformed to an always-on multi-tasking, asynchronous method of communications, easily switching between e-mail, text, voice, video between multiple people, often simultaneously depending on our communication needs or location or connectivity. Our communications has also gotten much more contextual, often adopting fine-grained preferences for how we communicate with a particular person in a particular social context.

There is no question traditional views of communications services are quickly being challenged because of this new world. These new capabilities along with the evolution of mobile and web applications and new application frameworks and technologies like HTML5 have opened the flood gates both for application developers and end users.

What does that mean for the communications service provider? What should a telephony product look like in the next 5-10 years and how does the communications architecture evolve to support it? What role can/should the service provider play in this new world of expanding communications capabilities?

Dreaming of the Next Generation Network

IP telecommunications has evolved to become mainstream rapidly in just the past 10-15 years since it was adopted widely. The PSTN or POTS network that VoIP has been employed to carry, in many respects, looks very much the same as how it looked over 30-40 years ago. Other than the conversion from

SS7/TDM to IP protocols like SIP, the core telephone service has not changed. There have been many attempts in various industry and standards bodies at extending and evolving the telecommunications network by defining a "next generation network". To a large extent, these efforts have not pushed the ball forward. The evolution of the Internet and interconnected IP network has allowed for delivery of voice and video in better, converged, and more efficient and cost effective ways. The building blocks for adding new and web connected "telephony features" has evolved as well. However, the fundamental black phone and dial pad interface, while important for basic communications capabilities, has been somewhat of a boat-anchor for the evolution of how the products around communications are defined.

Voice telephony is still THE primary service and still the basis of a multi-billion dollar industry; who's to argue for changing the formula?

WebRTC, What's new?

Enter WebRTC. By itself, some may argue that WebRTC is purely an API to access a camera/microphone, create a real-time channel, and display media on a remote client. What's new? There is a lot of push back from the telephony industry saying, WebRTC is just a new client in a crowded space of VoIP protocols that have existed for years. The potential for a revolution may be subtle depending on the viewpoint, but profound if put in the right context. When you view the capabilities of WebRTC in the context of a browser, all of the power and complexity of VoIP all of the sudden becomes a few lines of JavaScript code, a tiny component in the mix of a lot of other application and multimedia capabilities. It looks to be a stark new reality for the telephony product manager, but is it really a much larger opportunity in disguise?

Real-Time Application Architecture

WebRTC provides a flexible API to wrap real-time communications into new and different applications in a standard way. Interestingly, and often confusingly, WebRTC makes no assumption about signaling protocols or any definition around any specific application or service. WebRTC, at it's core, only concerns itself with the capture and display of media and the mechanics of setting up an IP channel to transmit the real-time media data.

To some, having no specified signaling or application protocol leaves WebRTC so open ended, it can be hard to wrap one's head around building any common framework to handle communications services, particularly in a generic enough way that a service provider can play any specific role. Put in a new context, however, if we embrace the limited scope of media session establishment, the world of all communications related applications, regardless of signaling protocol or functionality becomes accessible and adaptable. And isn't that really the better long term approach anyway?

Application Models

There are two classes of applications WebRTC was designed to support. As defined in [1], they are the triangular and trapezoidal models. The triangular model, shown in Figure 1, defines the case where clients that want to establish a real-time channel between each other, talk to a common central server that coordinates passing IP address contact information between each client so they can establish a media channel directly between them.

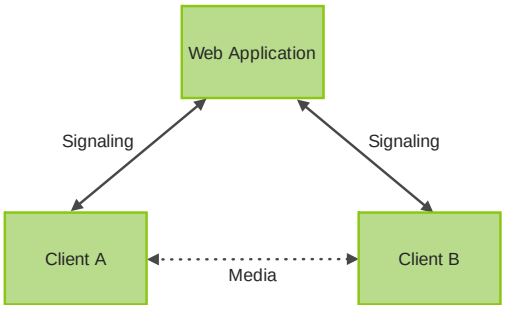

Figure 1: Triangle Application Model

The trapezoidal model, shown in Figure 2, defines the model where a client talks to a server associated with his particular application server, a remote client talks to her server, and there is an agreed upon signaling protocol and IP channel between the servers that passes the IP address contact information.

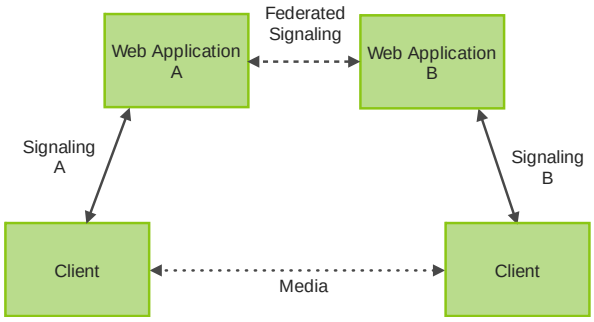

Figure 2: Trapezoid Application Model

This can be looked at as the federated model. "A" has a communications application with a particular signaling protocol, "B" has another communications application with another signaling protocol. So for a client of server "A" to talk to a client of server "B", there has to be a federation protocol that is agreed upon, to make the end-to-end communications work. There are explicitly 3 distinct legs of signaling in the trapezoid model. Both the triangular and trapezoidal models define virtually the entire universe of models of real-time communications applications.

These application models are important to help guide a general architecture that can flexibly support both models depending on the needs of the end application utilizing a real-time communications service framework. But these signaling models alone only go so far in defining what is needed to make an end-to-end service architecture real.

END-TO-END SERVICE ARCHITECTURE

Applications and services comprise components that must be delivered, secured, billed, managed, authenticated, and authorized. These requirements are common across services and can be abstracted into a generic service architecture. One such service architecture currently deployed in both cable and mobile networks is the IP Multimedia Subsystem (IMS), shown in Figure 3. The 3GPP defines the IMS specifications [2] for mobile providers; CableLabs, in turn, adopted and enhanced those specifications as the core of its PacketCable 2.0 initiative [3]. IMS defines an industry standard way for managing, billing, authentication, as well as the protocols and components used from an end-to-end perspective. IMS utilizes SIP and DIAMETER and other IETF defined protocols to deliver telephony services. SIP and IMS were born out of a specific signaling and service delivery model to support telephony services and the application and feature servers that can extend those services.

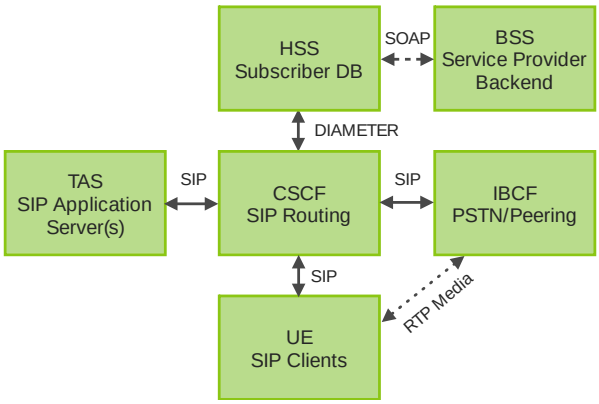

Figure 3: IMS High-Level Architecture

Comcast, as an example, has moved from distributed soft switch architecture as defined in PacketCable 1.0 and 1.5 [4] to the IMS/PacketCable 2.0 architecture. We have achieved many of the cost, operational, reliability benefits of moving to the centralized and highly scalable model that IMS offers. This is a clear win from that perspective.

However, as web-based services became more focused around building cross-platform and cross-service applications, and as newer web authentication and mobile applications models evolved, even as recently as in the past 1-2 years, there developed a clear need for a better service framework layer.

Because IMS derives its application model from the SIP family of standards and derives its service delivery foundation from SIP based deployment and management practices, there is an implied model that is hard to morph to support a general application framework. In addition, its heritage as part of the 3GPP and focus on a mobile vision of a single primary device per customer presented challenges, as will be discussed more in depth later. Extending beyond the concept of a managed device and managed primary singular identity was challenging. Incorporating non-telephony devices and services and even bring-your-own-device models, which can be supported for some cases, became burdensome to support in a general way.

These issues became the main motivation for the work presented in this paper. We propose a new telecom stack, one with the flexibility to support both traditional and non-traditional telephony servicesm and with a focus on the fundamental establishment of real-time communications channels through a flexible, web API-centric model. Just as WebRTC is designed without any specific signaling protocol, we believe being non-prescriptive to any specific service or application architecture, device management model or identity or set of identities is an approach that provides maximum flexibility in most contexts.

Identity

Many service providers, like Comcast, are not only telephony service providers, but also a provider of many services such as Internet and television and associated web services to supplement these services. They generally provide two main types of identities that can be associated with a particular account holder and user. These include an e-mail identity and a telephone number. Most web applications and services require an email address and user-generated password for account authentication, typically enforced by a centralized, application independent single-sign-on system (SSO). In the past few years, it is becoming increasingly popular, and for some services mandatory, required to use OAuth style token-based authentication. OAuth 2.0 [5], supports a common framework for authentication of user ids from multiple client environments, such as, mobile, browser, embedded device to name a few. This allows a single set of credentials to be managed by end users to access all of their service provider services on many different platforms with a single set of credentials managed from a central secure service.

Additionally, it is increasingly common for application and service providers to be agnostic about supporting both identities owned directly by the provider as well as those originating from third parties. These third-party identities, can be supported either by direct association of a user defined password to that third-party identity, or via what's commonly referred to as a three-legged authentication. Here, the application or service provider trusts the authentication token provided by what is usually an OAuth supporting third party identity provider (IdP). Popular examples include Google, Facebook,

Twitter Auth or more general industry initiatives like OpenID. This idea, in the context of WebRTC is detailed in [6], but this framework is now very commonly used in web applications in general.

As is common for many service frameworks born before these flexible identity authentication frameworks became commonly used, IMS and SIP generally assume an identity model where a particular identity type, a SIP URI or TEL URI, is used. Many SIP networks employ a specific set of credentials for SIP services, separate and distinct from any web based login credentials.

For authentication, IMS currently specifies either a SIM mechanism for the mobile terminal world, where a physical card with an embedded certificate/public identity is provisioned for an account, or in the case of many fixed line service devices like Cable E-DVA's and enterprise PBXs, SIP Digest based username/password authentication is used. The identity or telephone number, known in IMS as the public identity, is associated with the device authentication credentials, known as a private identity, are both stored in the subscriber database, HSS.

From many perspectives, having multiple credentials associated with a user is both difficult to manage, inconvenient for the end user and can be a security risk.

Service/Device/Identity Abstraction

How do we provide the flexibility needed by the web and provide a service that can be managed in a reasonable way? The web model fundamentally a distributed, abstracted model for services, applications, devices, and identity. A service provider wants a common way to offer services to other services, applications, devices and using an identity of the users choice. The question is how do we move the telecom stack to adopt these principles in a way that is consistent with this level of abstraction.

Our proposed solution conceptually started as an extension of the routing model that is core to IMS and SIP services and incorporated a similar user/device based model and architecture with a web services style abstraction. While this work started before WebRTC was born, it quickly became quite intentional to adopt many of the ideas from the WebRTC and general web framework. These concepts include:

- Identity - support the ability for end applications to use any identity model, either self managed, third party managed, or service provider managed.

- Device - allow for the ability to support multiple devices, either simultaneously, independently, and dynamically without any dependency on provisioning or management interface

- Application - support either the triangular or trapezoidal application models with or without the dependency on service provider routing services (i.e. SIP/PSTN/RCS/federated routing models)

The model was defined as a single API to support both internal and external applications, with a primary focus on the ability to setup of real-time media channels. The resulting core service became very concise and clear, a media session establishment API, with hooks to specific traditional routing services and additionally some value added media functions as well (e.g. mixing, transcoding, etc.). This API additionally, depending on calling identity and domain, can be used to establish sessions between end clients directly, over PSTN, over SIP peering, or other federated models that may appear in the future.

In the future, this framework can be extended to new models of real-time

communications services that don't exist today.

PROPOSED ARCHITECTURE

The service framework we propose in this paper began as an extended SIP framework, supporting mobile applications as an extention to primary-line IMS services. Shortly after WebRTC was first proposed in W3C and IETF the framework was quickly adapted and extended to incorporate a fully HTTP based flow utilizing technologies such as Websockets and OAuth to provide a flexible solution that can exist entirely in the HTTP world and as part of a hybrid HTTP/SIP model. It has developed both from the needs of our product evolution and new product requirements including the grander motivation to address all the issues stated earlier as a first class design criteria.

The high level architecture is shown in the figure. There are a few major architecture components, many existing and common network elements not specific to this architecture and some new components developed specifically to support general real-time communications and media session establishment aligned with WebRTC and rtcweb protocols.

The existing infrastructure utilized includes:

- IMS/SIP infrastructure - the main SIP based PSTN and federated service routing component, extendable to other SIP services including RCS and VoLTE interworking and other future interconnected federated services

- Notification Services - a common platform for events and notifications, supporting Comcast managed devices such as set-top boxes as well as federated notifications including iOS and Android notification services

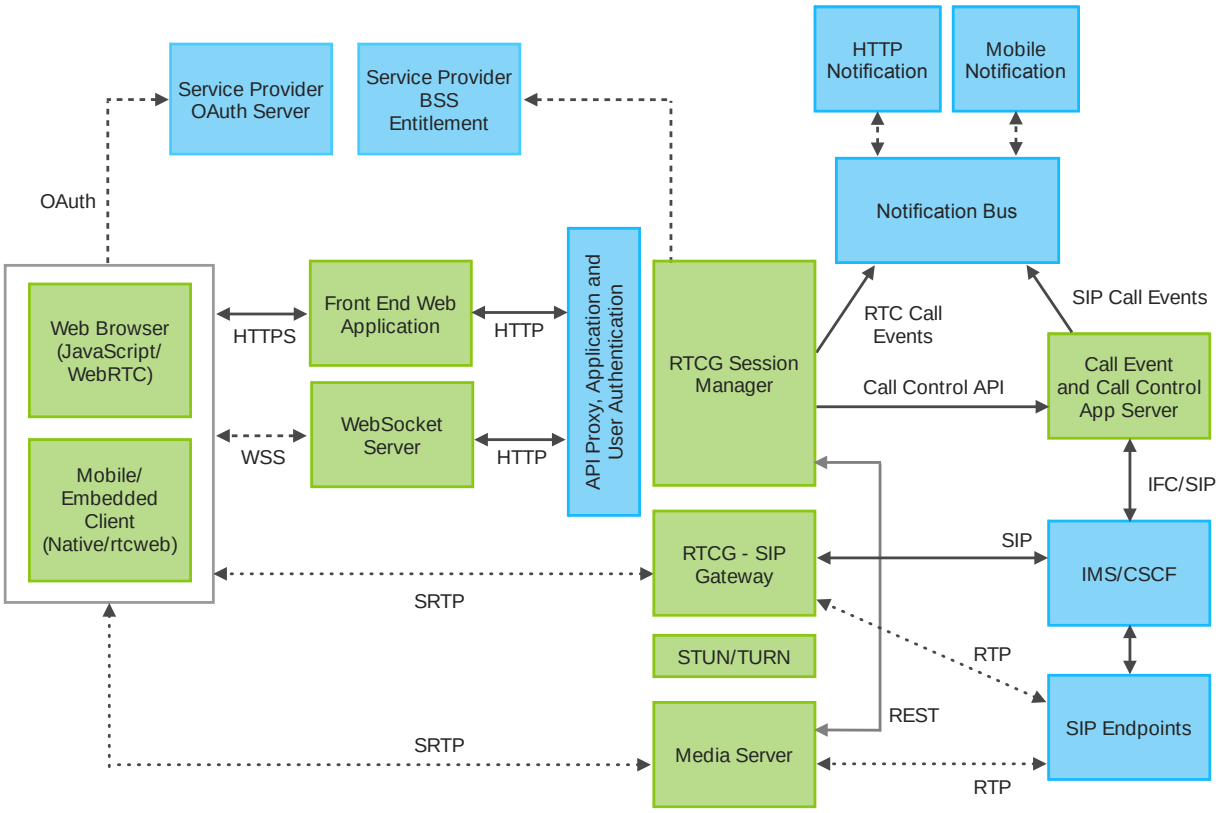

Figure 4: Proposed RTC Service Architecture

- API proxy and security layer - a common API proxy layer that incorporates application and user level authentication and typical API exposure and security functions

- SSO/OAuth Server - a common authentication service supporting OAuth token authentication for user IDs

The additional components that will be discussed in more detail include:

- WebRTC clients - including JavaScript based, browser and browser-like clients as well as native mobile SDK based clients supporting rtcweb protocols.

- Real-time Communications Gateway (RTCG) - includes a session manager, user manager, and SIP signaling and media gateway component to provide interworking with IMS/SIP and RTP media streams and codecs supported in the federated networks

- Application and WebSocket Server - represents the multitude of applications that can utilize RTCG services and provide a signaling path that is either application specific or conforms to a reference signaling model

- Call Event and Call Control - this component allows the bridging of incoming calls from federated networks to and from the RTCG

WebRTC clients

WebRTC is part of the W3C specifications and what some refer to as the HTML5 set of browser capabilities. It defines a set of Javascript APIs, including primarily getUserMedia and peerConnection. These APIs define both how to access the camera and microphone of the underlying OS platform as well as the establishment of a particular RTP based media channel between two peers. The API is defined in the W3C specifications, and the specific protocols are defined in IETF under the rtcweb working group.

We focus on two classes of clients, but aren't necessarily limited to these. The first is a JavaScript based client using the W3C defined WebRTC APIs that either supports a traditional third party browser application, or a more integrated device that provides a browser enabled environment in order to support "embedded" HTML5 applications. A JavaScript reference SDK is provided to support a particular signaling protocol based on websockets and incorporates the specifics around authentication of user and/or application. The other client model is a native approach, where an SDK native to the device is provided with either C or Java based APIs similar to the W3C APIs. These specific clients are typically in scope of most WebRTC based services today, but as stated, is not limited to these device or application models.

Real-time Communications Gateway

The RTCG is the network component handling basic registrar and media routing logic. It exposes a RESTful HTTP based API for establishing media sessions. It acts as a dynamic registrar and routing proxy and is for the most part agnostic to identity with optional configurable routing rules based on specific identity and domain. We very explicitly wanted a model that was dynamically able to support any unique identity in the context of a particular unique application id. Additionally, it was important to support this with minimal provisioning or configuration, if at all.

The main federation interface supported is SIP. This can be extended to other gateways

supporting protocols or even federating to other RTCG supporting service provider networks. In the case of SIP and IMS, a media session initiation is translated into a SIP INVITE to the IMS SIP network, via Mw interface to CSCF. The SIP REGISTER method is specifically not supported. In the terminating case, an INVITE toward RTCG SIP Gateway is translated into a media session on the RTCG. Otherwise, both signaling and media are handled very similar to traditional signaling and media gateway components. In the case of SIP to WebRTC, media is likely converted between RTP and SRTP/DTLS and if transcoding is necessary, it can be incorporated. ICE and TURN procedures are followed for WebRTC clients and the exchange of credentials for TURN is handled in the API.

Another important change to note in the architecture is around the use of notifications as the primary mechanism to signal the initiation of a potential session. In the traditional SIP architecture, there is an explicit REGISTER method that provides a persistent connection to a SIP registrar that allows an INVITE to be sent to the associated contact address. We have moved away from this idea for a number of reasons:

- The number of potential registered devices is growing exponentially multiplied by the potentially exponentially growing number of applications
- Because of different network topologies and NAT and firewalls, holding persistent connections to devices can be a challenge
- Because many mobile devices are powered by batteries and persistent connections to networks can be very expensive in terms of power, it is an often discouraged practice
- Most important, the modern application interaction model has changed, web pages are not persistent, mobile applications can be persistent, but we interact with them in short sessions and based on notifications, rather than having an application always in the forground

That said, the dedicated telephone device model can be supported with a persistent registration model if needed, but we see this model less and less relevant as time progresses.

Application and WebSocket Server

One of the key design criteria for the proposed architecture is the separation of application and routing. The application interface should be a convenient API that hides the details of routing and session management from the application layer. The architecture shouldn't impose any assumptions or constraints around the application developer; there should be clear separation between application and RTCG. There also isn't any assumption around application environment. Any modern HTTP supporting server side development environment can be supported.

The minimum requirement is the application developer only needs an application key to authorize access to the RTCG APIs. Some applications that may require authenticated access to service provider services. For Comcast, as an example, this might include PSTN calling from the Comcast customer TN or access to specific paid services that require Comcast user credentials in the form of an OAuth token provided by Comcast SSO system. In this case, the application specific credentials along with the Comcast user token can be passed in the HTTP requests to RTCG and an entitlement check is performed in the RTCG to validate the association with the account specific authorized services. Another

example application might need the access to a media stream from a managed secure device such as an in-home security camera. The three-legged auth model can also potentially be employed to support both end-user directed auth and revoking of a third-party application to access to these types of media stream services.

Additionally, there is a provided reference client SDK, application server and websocket server to support a particular signaling model that can be used by application developers that don't want to build their own signaling.

Call Event and Call Control

Because of the notification-based mechanism that is imposed by the architecture, for incoming or terminating calls to a federated network like IMS, there needs to be a mechanism to send a common set of call events and provide an interface for third-party call control. The importance of notifications was discussed above, but the mechanism to report incoming INVITEs and other call state details allow the end WebRTC client to interact with the call in the federated network. The call control interface provides the mechanism for when the client wants to pull or push a call to/from the federated network in the same style third party call control works in SIP today.

CONCLUSION AND NEXT STEPS

Much of this work was part of an architectural evolution born out of the necessity of supporting a more web-oriented approach to communications. From the beginning, we intentionally kept our view very broad, from the basic ability to support OTT telephone soft clients to the ability and flexibility to support the potential universe of non-traditional real-time communications applications. The guidance of the fundamental principles of WebRTC combined with a fundamentally web-centric approach to integration into the service provider common services sets the stage for a truly new approach to the integration of real-time media services.

To the casual observer, media streaming over the Internet seems like a solved problem. It is common to see HD resolution video streamed over IP with generally high quality and latency. Of course, the important distinction of telephony types of real-time communications is that minimizing end-to-end delay is an important requirement. With stored or buffered live streaming timing constraints are very much relaxed, often in the order of seconds or 10s of seconds. Delays in the order of low hundreds of milliseconds or lower are critical to delivering a quality experience. Even today, this continues to be a sometimes difficult challenge over varying network conditions and topologies. Though aggregate network speeds have improved immensely, there are still existing bottlenecks that have plagued real-time communications over IP networks from early on. There are many efforts in the IETF, as an example, to specifically tackle these issues and look at minimizing congestion over and above traditional congestion control techniques and priority packet marking techniques. As a service provider, building a standard framework that enables a more predictable experience for its subscribers across all of the applications they use can be an interesting benefit to employing standard APIs for media stream management. There have been various attempts at this in the past, but perhaps WebRTC and the proposed architectural framework can be the technology to rally around to deliver it in a consistent way.

Extending the Framework

For those familiar with WebRTC, one perhaps glaring omission from this proposal is regarding non-media related real-time streams such as the WebRTC data-channel or even websocket channels. It doesn't take much

imagination to recognize that a very similar framework can be employed to support certain real-time data classes of applications including gaming, messaging, machine-to-machine or Internet-of-things types of use cases.

We would like to get industry and community feedback around this proposed framework. Not unlike IMS, we believe there is a large opportunity to evolve real-time communications as a consistent framework for either application integration or federation of services, even beyond PSTN type services. With a specific focus on a flexible web based API for media session establishment alone without any application signaling assumptions, this framework is much better positioned to evolve the telecom network to a new generation of applications and services.

ACKNOWLEDGEMENTS

I would personally like to acknowledge and thank Bryan Paluch for his collaboration and work in taking this from early concept to implementation and his valuable contributions to refine and extend this proposal to the state it is today. I'd like to thank, Erami Botchway, John Hart, and Bryan Paluch for their valuable comments and review. I'd also like to thank our product and business teams that had the vision to see the potential in being a bit disruptive to forge new ground and test the waters of new opportunities in next generation communications.

REFERENCES

[1] draft-ietf-rtcweb-overview
[2] 3GPP TS 23.228 IP Multimedia Subsystem Stage 2
[3] CableLabs PacketCable Architecture Framework PKT-TR-ARCH-FRM
[4] CableLabs PacketCable 1.5 Architecture Framework Technical Report PKT-TR-ARCH1.5
[5] draft-ietf-oauth-v2
[6] draft-ietf-rtcweb-security-arch