A NOVEL SCALABLE ARCHITECTURE ON ANDROID TO CATER END-2-END IMS SERVICES TO AUDIO VIDEO TERMINALS

Suman Kumar S.P[1] and Vijay Anand[2]

[1]Centre of Excellence (CoE) Division, Aricent Communications, Sigma Tech Park
Suman.Prasanna@aricent.com
[2]Whitefield Main Road, Bangalore-560066.Karnataka, India
Vijay.Anand@aricent.com

ABSTRACT

IP Multimedia Subsystem (IMS) known for its prominent service delivery is paving way for applications like Video Voice over IP (VVOIP) on handheld devices too. Android, a software platform from Google, pioneering in middleware and applications domain of handheld devices does not offer support for IMS services by design.

Motivated by these observations, we present novel scalable architecture to cater IMS based services on Android. First, we discuss few related works to investigate and analyze the IMS and the role of SIP in IMS services. Second, we discuss few characteristics of android architecture and detail constraints associated with middleware sub-system to cater IMS services. Consequently, we describe the proposed architecture addressing inherent challenges like QoS, Extendibility and Security encountered while offering IMS based services. The architectural framework design reduces time-to-deployment by using a layered approach encompassing three layers, namely: ***QoE Engine, Advanced IMS Service Core (A-IMS) and Application Services Engine (ASE)****. The A-IMS core is responsible to provide End-2-End efficient service delivery of IMS services like VOIP. We point out the benefits of using the ASE and generic media management and control interface (GMMCI) to ensure smooth and swift portability onto different flavors of android. Finally, we conclude by detailing some use-cases of future IMS services like Video Sharing/Video Recording in an on-going Video Telephony Call.*

KEYWORDS

Android, IMS, Middleware, Application Framework, Wireless Network.

1. INTRODUCTION

Traditional telephony comprised of POTS lines which were incapable of supporting services at the end-point, also had a complete dependency on the network side components. On the contrary, with the rapid advancements in terms of computing & power at end-points (like IP Phones), the intelligence is progressively moving towards the network edge. SIP is an application-layer session control protocol which has been designed keeping the above requirement in mind, embeds a lot of call-processing intelligence in the end-points. On one hand, SIP is the driving technology behind the telecom industry's shift from circuit switched TDM based infrastructure to the more flexible and web-centric packet network infrastructure of IP telephony. Its flexibility and openness have made it a critical building block for the creation of innovative enhanced services, such as voice and video messaging, conferencing and prepaid calling. With growing adoption of packet communications, service providers are transitioning from a voice-centric orientation; the connection and delivery of voice calls; to a service-centric model focused on the rapid efficient delivery of innovative services to consumers [1, 2] as shown in fig 1. Although there are many competing visions for how service delivery can be

DOI : 10.5121/ijngn.2010.2306

achieved in next-generation networks, the IMS or IP Multimedia Subsystem, with SIP as an integral part is currently gaining considerable momentum [3, 4]. On the other hand, Android, a software platform from Google has pioneered to cater middleware and applications for handheld devices. This is precisely due to its ability to offer Appealing Entertainment experience for Value Added Services (VAS) on resource constrained handheld devices [5]. Android employs Packet Video Multimedia Framework (PVMF) to cater media related services. The PVMF has two engines: PV Player Engine and PV Author Engine. These engines which were designed to cater basic functionalities of Media Playback and Recording inherit one major challenge of adaptability to cater advanced VAS like VVOIP. This shortcoming in the android middleware sub-system has motivated us to design an extendible robust architecture on android to offer various IMS based services.

Figure 1: Real Time Scenario

In this paper, we propose novel scalable software architecture to cater IMS based services on Android. The architecture tends to futuristic by using a "Module Interconnection Architecture" design to provide the required flexibility to add new features. The layered approach employs logical interfaces among the defined engines to ensure functionality based decoupling for easier portability. This paper also provides different architectural views like "Meta Architecture", "Conceptual View" and "Logical View" to stimulate further research in this direction. In an attempt to demonstrate the evaluation of this architecture, we have demonstrated few common user scenarios along with implementation guidelines like MSC (Message State Chart). We conclude by providing various use cases with actual sequence diagrams to aid users adopting this architecture.

2. DESIGN APPROACH

In this paper, we envision to provide "The Big Picture" of the system architecture of android to cater IMS services. We discuss few concepts about signaling and media flow interaction within IMS. We also detail the architecture aspects of Android and their usage in such a system. Most of the research work has explored the issues surrounding the SIP-based model for control model and delivery [6]. Inline with these research observations, the proposed architecture embodies an Advanced IMS Core (A-IMS Core) layer to control and manage signaling functionalities. It is also responsible for smooth handover to media flow components upon successful signal activation. Media services being realized through a series of functional elements in the IMS; real-world implementation experience with SIP has highlighted its limitations to offer efficient media delivery like Voice quality, Acoustic Echo Cancellation (AEC) and so on. To address this issue, our proposed architecture has a dedicated QoE Engine which interacts with A-IMS Service Core and GMMCI to enrich user experience for real-time applications like VVOIP. The design of android architecture has two notable advantages: First, the communication among layered components is asynchronous in nature. Second, the

functionality based layer classification to offer their services to the applications. This architecture design goal is leveraged by the middleware components (A-IMS, QoE Engine) of our architecture. Also, Android architecture is based upon an opaque IPC model. Applications, Services expose their functionalities to the system, and at runtime, other applications can request these functionalities. Essentially, the platform provides a managed and secured late code binding. This model is specifically useful in the interaction between Application Services Engine and Android Application framework to ensure easier portability. Section III illustrates the proposed software architecture along with brief description and functionalities of the associated software modules.

3. PROPOSED SOFTWARE ARCHITECTURE

Recent Studies indicate that consumer interests in usage of IMS services are moving from user-oriented to group oriented communication. OEM's are now tending to offer combinational services for media-centric applications like Video Conferencing on handheld devices [7]. These applications which demand very high End-2-End efficient media delivery is still a major challenge on resource constrained devices. In line with these objectives for next generation IMS services, the proposed architecture (shown in fig. 2) principally covers media-centric services and applications extending the android software platform with following components:

a. **IMS Applications:** Voice over IP (VOIP), Video Voice over IP (VVOIP), Video Share and so on.

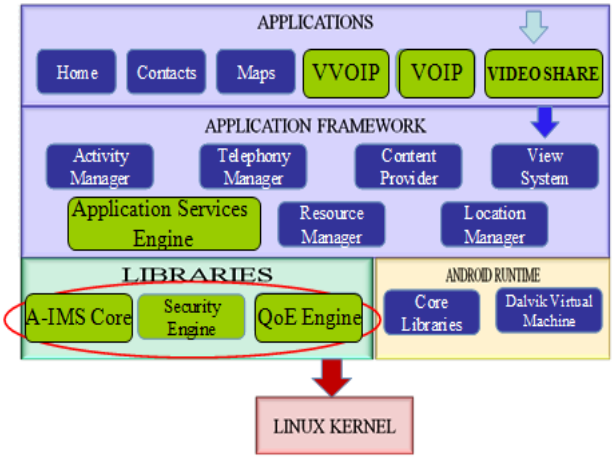

Figure 2: Proposed Software Architecture

b. **Advanced IMS Service Core:** The SIP protocol chosen by IMS architecture brings in a new set of challenges in wireless domain due to its in-effective session management. So, a need was felt to evolve IMS architectures to manage interactions between SIP and non-SIP applications. The A-IMS Service Core provides functionality to cater services like End-to-End VOIP.

c. **QoE Engine:** It is primarily responsible to discover optimal values for key parameters like bandwidth, for different kind of applications. It is responsible to provide core QoS functionalities like Audio/Video Sync, QoS Management, Media Management, Echo cancellation, Noise Suppression, Packet Handler and more.

d. **Application Services Engine (ASE):** The Application Services Engine is responsible for responding to events posted by middleware and interacting with Android Application

framework components. The ASF reacts with input and external events, which includes User input, Connection & application status for IMS applications.

4. ARCHITECTURAL DESIGN ASPECTS

The proposed architecture (shown in fig 3) intensively focuses on QoE, loose coupling and extendibility as three parameters impacting the overall system offering key IMS services [8, 9]. The A-IMS Service Core and the QoE are primarily responsible to handle the complexities of the IMS network architecture to provide better voice quality and reduce voice latency. Another focus point being the end-to-end security, the Mobile Security Engine (MSE) of our architecture caters to this requirement.

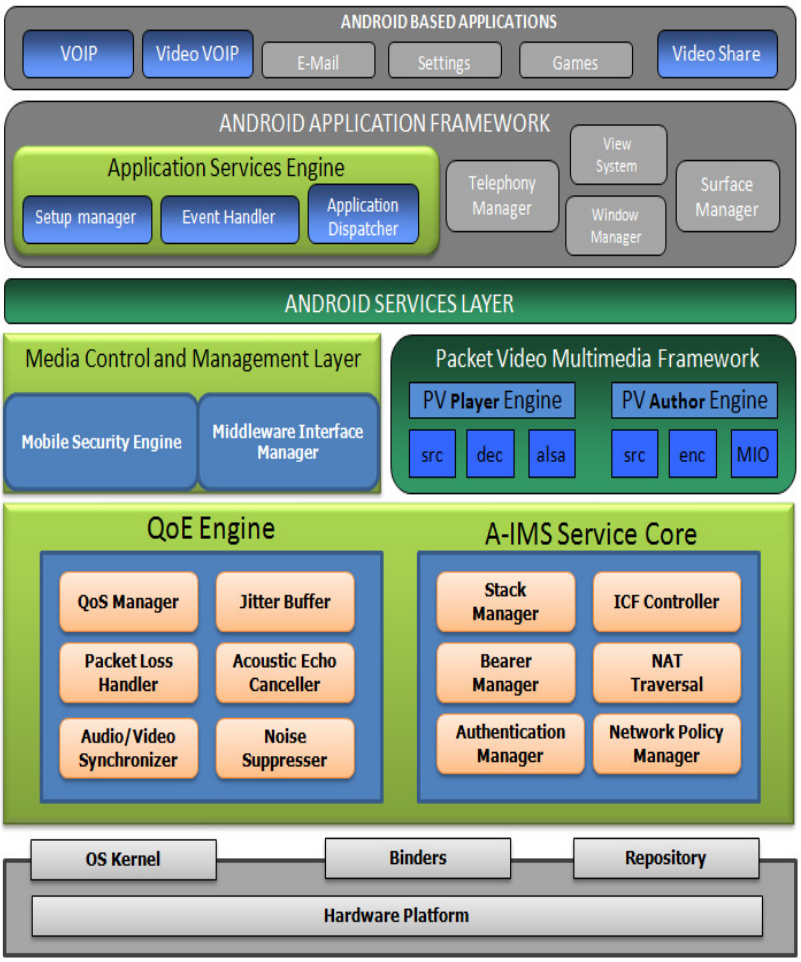

Figure 3: Conceptual Architecture View

4.1 QoE Engine

It is the heart of the architecture emphasizing the need to provide better QoS to real-time applications like VVOIP. It is composed of various modules providing key functionalities like QoS, Jitter Buffer Management, Packet Loss and lip-sync to name a few. Every application developed gets connected to one or more modules in this layer depending on the nature and usage of that application. Ex: A VOIP application gets connected to noise suppression module to provide a better voice quality. Another application like VVOIP might get connected to all three modules (QoS, AV sync handler, Network packet loss handler).

QoS Manager: The QoS manager module is responsible for measurement and monitoring of QoS parameters using media protocols like Real Time Control Protocol (RTCP). The RTCP-XR expands base RTCP metrics to include:

- **Delay:** The maximum tolerated delay (ms)
- **Jitter :** The maximum tolerated jitter (ms) (the variation in delay)
- **Bandwidth:** The bandwidth reserved for a packet or a flow
- **Acceptable Rate**: The minimum acceptable rate (bit/sec)
- **BER/FER:** The Bit Error Rate (BER) or Frame Error Rate (FER)
- Signal and Noise level
- Packet discard rate (at RX jitter buffer)
- Jitter buffer size and configuration.
- Packet loss that is transitory, consecutive or sustained

This module is mainly involved to monitor the session quality by tracking key QoS parameters in an on-going multimedia session. The job of QoS Manager is to store the predefined QoS attributes in the information table, monitor and compare the negotiated QoS parameters with the actual such that if a change in the negotiated value occurs, then initiate the trigger to user to know the current value of QoS.

Design approach of QoS Manager: The design approach which meets QoS Requirements requires satisfying the following four objectives:

- **Objective 1:** Maximizing user device preferences.
- **Objective 2:** Maximizing application bandwidth
- **Objective 3:** Minimizing Jitter, delay and Loss
- **Objective 4:** Minimizing bandwidth fluctuations

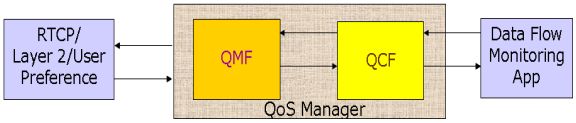

Figure 4: QoS Manager Software Modules

- The Source or the user specifies the values of different QoS parameters with their minimum/maximum values of tolerance.
- The QoS parameters along with their respective values are given to the QoS Manager during run time.
- If QoS Manager receives QoS parameters for a packet of the flow for the first time, it sorts the parameters according to their relative importance. After, that the QoS Manager calls a specific functionality to take care of the parameter that is relatively the most important. After that it takes appropriate steps to take care of the next relatively important parameter (if possible), and so on.
- The information is then delivered to the UI according to its QoS specifications and the source is informed accordingly.

If the source is not satisfied with the QoS of the packet delivered, it informs the QoS Manager about the change it wishes to have in the QoS. The QoS Manager tries to adjust the QoS parameters accordingly.

The QoS management functional entities as shown in fig 4 consist of:

- **QMF (QoS Measurement Function):** The QMF entity is responsible for collecting all the information about the QoS Parameters of on-going media session using the RTCP and passes the same to QCF (Quality Control Function).

- **QCF (Quality Control Function):** The QCF entity receives information from the QMF for renegotiation of the QoS parameters receives the QoS criteria of the existing link, compares negotiated QoS parameters with the same link and checks whether or not the QoS criteria for the existing service are fulfilled. In such an use case, the application/user specifies the values of QoS parameters, their minimum and/or maximum values, relative importance, and tolerance limit for each parameter. As mentioned above, QoS Manager sorts the parameters according to their relative importance. The QoS Manager interfaces with appropriate modules (Stack Manager, ICF) for providing the QoS.

Jitter Buffer: This module is primarily responsible to remove the effects of jitter from the stream, by buffering each arriving packet for a short interval before playing it out. This also suffices to substitute additional delay or packet loss that might be seen at receiving end. A fixed jitter buffer maintains a constant size whereas an adaptive jitter buffer has the capability of adjusting its size dynamically in order to optimize the delay/discard tradeoff. Both fixed and adaptive jitter buffers are capable of automatically adjusting to changes in delay. Adaptive jitter buffers reacts to either discard events or measured increase in jitter level. When a discard event is detected then the jitter buffer size is increased and when there is no discard event then buffer size is reduced.

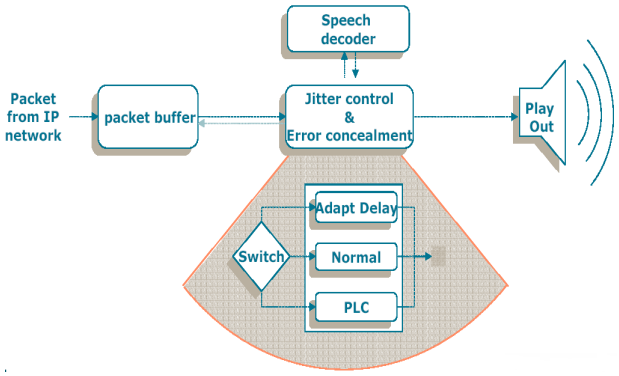

Figure 5: Jitter Buffer Module – Interface Diagram

Network Packet Loss Handler: Upon successful call establishment, the media data communications happens in Real Time Protocol (RTP) format. The embodied Real Time Control Protocol (RTCP) protocol provides periodic reports (RTCP-SR, RTCP-RR) which detail information on packet loss/corrupted packets of the on-going session. If any media packet is lost (may be discarded by the component or never reached), this module takes appropriate action as follows: -

- If it is a VOIP application, any audio packet which is detected /declared as lost by the protocol component, it will insert silence frames for the missing audio packets.
- If it is a VVOIP/Video Share, any video frame which is detected /declared lost or arrived late, the frame will be discarded and the display/render will render the next appropriate video frame.

Audio Video Synchronization: This module is mainly responsible to ensure synchronizing audio and video streams when rendered simultaneously. It interacts with the Jitter Buffer to fetch the data. The following are sequence of operation that occurs to achieve the synchronization. From time to time, this module synchronizes the video clock with the current audio time. This ensures that the rendered video is always synchronized with the currently rendered audio data. In general, the audio is used as a reference clock. However, this module also has the intelligence to make the audio clock wait in-order to catch up with the video data. Based on the application type and QoS statistics, appropriate method is used within the module.

- Audio rendering shall be the master clock for synchronizing the video. This clock shall be derived from the number of audio samples played at any point of time.

- The engine calculates the current audio rendered time based on the amount of audio received at NAL.
- The sink/renderer on its part maintains a clock of its own clock, depending on which the video frames are rendered.

Receive Side Video Rendering

This section describes the logic used for rendering the video frames using the presentation timestamps extracted from RTP header information of each frame. The first frame is rendered as soon as it is decoded. This system time at this point is taken as the reference time. Let this be Tr.

- Let Td be the system time when a subsequent decoded frame is available.
- Let Tp be the presentation time of the frame decoded at instance Td. This time is relative to the first frame's timestamp in media time units
- If (Td – Tr) == Tp, render the frame immediately
- If (Td – Tr) < Tp, set a timer for (Tr – Td - Te) where Te is the approximate overhead for firing of the timer. In the timeout function, render the frame without any checks
- If (Td – Tr) > Tp, frame is delayed for display. This condition should not occur if all performance parameters are met.

Send Side Video Frame Timestamp Generation

This section describes the logic used for timestamp creation to embed in the RTP headers.
Based on the frame capture rate configuration, calculate the frame time stamp whenever a frame is available from the camera. For the first frame the time stamp shall be 0. For subsequent frames, Let 'F' be the capture rate in frame/second.
Let Tr be the frame time stamp which will be calculated as (1/F) * 1000 milliseconds
Current frame time Tf = Tr + Ta where Ta is initially 0, and Ta = Tf after every calculation for a frame i.e., it contains the accumulated time.

4.2 Advanced IMS Service Core

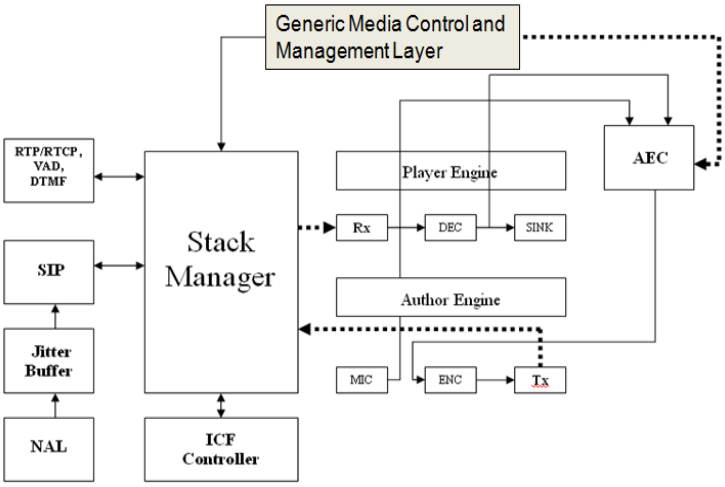

Figure 6: Execution Architecture for VOIP on Android

The above fig 6 shows the execution architecture of VOIP application using the described modules. The Network Abstraction Layer (NAL) and Stack run on independent threads. Whenever the data is received on NAL thread, the data is pushed to the Jitter Buffer and a notification is sent to the Stack Manager. Stack Manager which is running on a different thread, will read this data and invoke corresponding media protocol stacks (RTP/DTMF/VAD) to

extract the encoded data from the received packet. Upon Success, stack manager intimates the GMMCI to provide these encoded data to the PVMF nodes to decode and render the audio samples. The QoE engine components also play a vital role to ensure better voice quality by performing algorithms like Acoustic Echo Cancellation (AEC) on the encoded data.

Bearer Manager (BM): This module allocates resources and manages bearer traffic to meet customers' service quality requirements. The primary functions include policy enforcement, mobility management, security, accounting, and access control.

ICF Controller: It is responsible for communicating with Stack Manager, GMMCI to provide the routing and other network information to various applications. ICF just tunnels the POC/Audio-Video app/configuration requests from the applications to GMMCI and the response from GMMCI to corresponding applications.

Stack Manager: This module is primarily responsible to control and manage signaling and media stack components. It is composed of SIP Stack, RTP/RTCP Stack, VAD/CNG and DTMF Stacks. These protocol stacks are passive components. The stack manager is an active critical component controlling both signaling and media flow. It interacts with GMMCI and Jitter Buffer of QoE Engine to provide data to the PVMF framework.

NAT Traversal: This module implements the Network Address Translation (NAT) traversal features that allow calls to be placed and received from behind routers that implement NAT. It is pre-integrated and co-operate directly with the media processing components of the PVMF, eliminating the need for additional configuration by in-house development teams [10]. This has the intelligence to identify which type of NAT is being used and allow the ICF Controller to specify the corresponding NAT Traversal protocol.

Policy Manager (PM): It is the primary decision point for network policies. i.e., deciding the ways on how the underlying network supports applications.

Authentication Manager (AM): The SIP service platform in the network that authorizes access to SIP services provides SIP registration and authentication functions, and is responsible for the invocation and management of SIP-based features. This module is an interface extending the above defined functionalities to BM.

4.3 Mobile Security Engine – MSE

A new Mobile Security Engine is presented as shown in fig. 7. It separates the development of security services from the application development. This allows application developers to seamlessly integrate tunable security services with their application. The main motivation of MSE is to provide the provisioning of services and secure access to user and application profiles [11, 12].

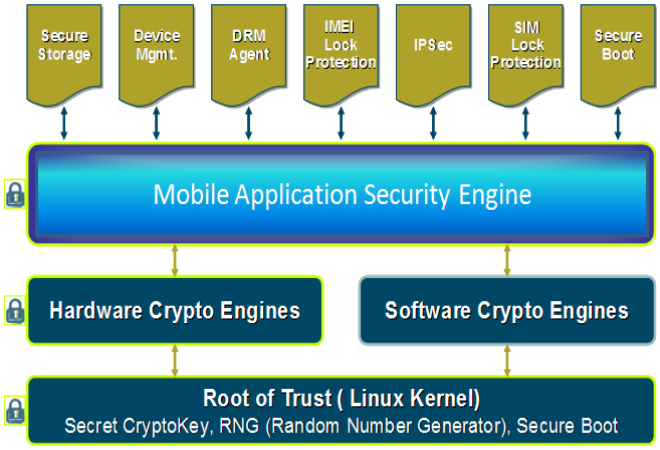

Figure 7: Mobile Security Engine

The main functionalities of the MSE described in Fig 7 are:

- It separates the development of security services from the application development.
- It partitions security services into different security levels so that appropriate security services are deployed to best trade of the needs of security and performance preferences.
- It allows application developers to seamlessly integrate tunable security services with their application.
- It is a stand-alone security middleware entity providing services and functions such as key generation, encryption and decryption with private key etc.,
- It provides facilities that could be integrated with DRM Ex: Device authentication.
- It is aware of different security facilities and can adjust security levels according to the security requirements.
- It also provides security policies that include mechanisms for assurance of mutual authentication, authorization, integrity, and confidentiality.
- Furthermore, this layer makes the integration of security into the existing mobile application framework, a more realistic and conceptually feasible task.

4.4 Generic Media Management and Control Interface (GMMCI)

The PVMF of android has three engine cores: PV Author Engine, PV Player Engine and PV 2-way engine. IMS based services giving rise to range of applications like VOIP, VVOIP, Video Share, DLNA, MTP and IPTV and so on; each such application requires the Multimedia Framework to be integrated with the system. The 2-way engine might be used only for Video Telephony (VT) kind of applications while Player and Author Engines might be needed for applications like VOIP/VVOIP and the like. Application Components needs some level of abstraction between media processing elements and other components of android.

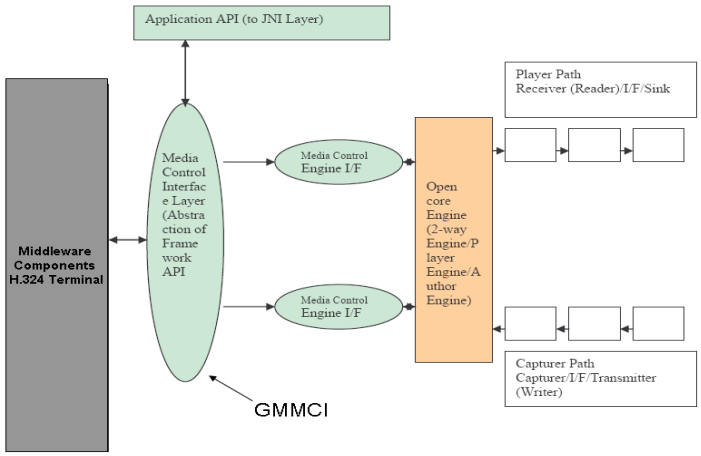

Figure 8: Generic Media Management and Control Interface

The Generic Media Management and Control Interface (GMMCI) of the architecture (as shown in fig 8) cater to such a requirement for easier portability. This interface will be separated with the application and its engine graph creation, control. This layer will control the PVMF and decouples the control flow (signalling flow) for VOIP kind of applications. This allows the integration of the application with the existing framework smoothly and swiftly. The Open core component of android might support more than one framework (like vorbis) in which case, GMMCI is the only component that needs to be modified i.e., applications are independent on the underlying multimedia framework. This generic design also caters to future applications like DLNA which might be a result of technological convergence. In such a case, the GMMCI abstracts the DLNA specific API and underlying media sub-system.

4.5 Application Services Engine

Event Handler Module: The event handler handles the following events from external applications.

- **Incoming Audio call / Video call**
- **Notifications from Multimedia applications**

The Event handler module is responsible to provide suitable notifications to the UI components. In a typical local playback case, the audio/video rendering is stopped because of other higher priority tasks (for e.g. incoming call etc.) is being noticed in a system. On receiving such a notification, the Event Handler switches to the running state to continue the VOIP call and notifies the user through specific interface. When the audio resource is again available, Event Handler informs the respective application component through the Call-back (Notification) interface. The user can then resume the file playback.

Application Dispatcher Module (ADM): This module is responsible to process the middleware and internally generated events and invokes corresponding applications. It receives events through GMMCI and corresponds to respective application components. Ex: In supporting Voice mail feature of VOIP, the Stack Manager will invoke the DTMF stack and provides the encoded data to the QoE Engine. The QoE engine will further add it into Jitter Buffer and notify the GMMCI about the status. The GMMCI will in-turn invokes the ADM to provide an indication to the user about the arrival of voice mail.

5. USECASE SCENARIOS

The diagram in fig. 9 shows the logical architecture view of the system components when a VVOIP application is launched.

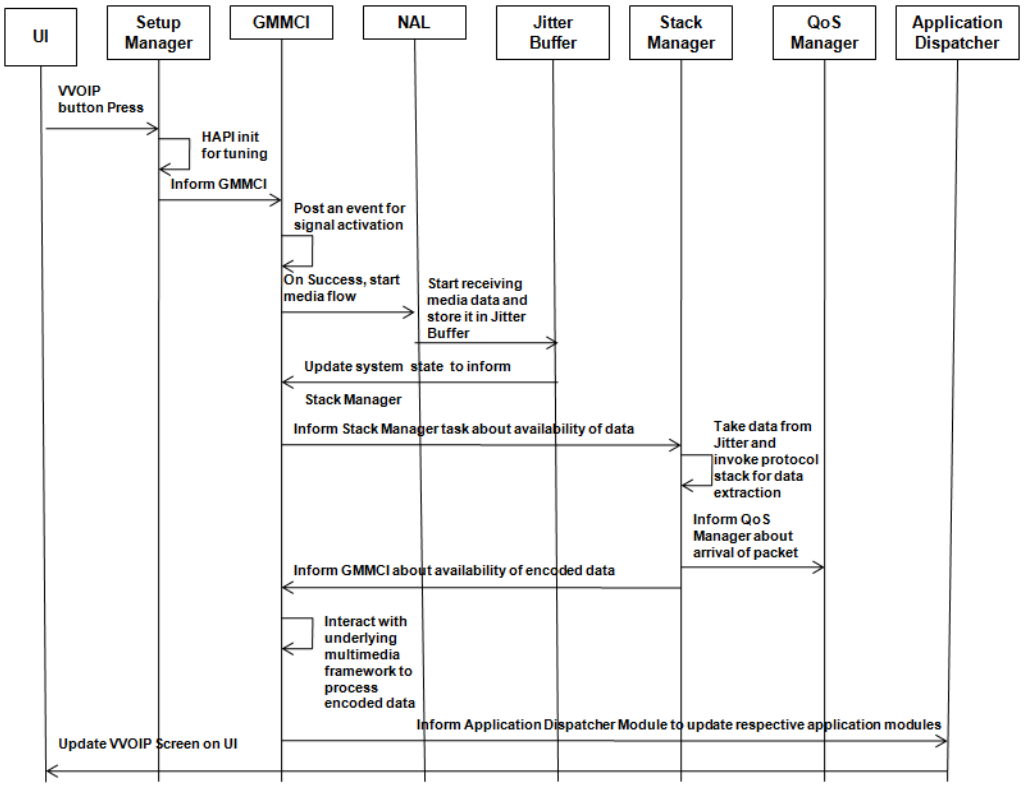

Figure 9: Logical Architecture View

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed novel scalable service architecture to cater IMS based services on Android. We detailed few concepts of SIP and the constraints associated in middleware subsystem of android to cater IMS services on Android. Consequently, we described the characteristics of the proposed architecture addressing key inherent challenges like Extendibility, QoE and Security. We outlined and pointed the need for Advanced IMS Service Core to cater End-2-End service delivery of next generation IMS based services. Also, robust media management and service delivery being essential to reconcile the characteristics among handheld devices, a dedicated QoE engine employed within framework accommodates various real-time IMS services like Video Voice over IP (VVOIP) on android. We showed the efficient design methodologies employed within the architecture to ensure easier extendibility for wide spectrum of applications. Another point of focus being end-to-end security, the Mobile Security Engine (MSE) of our architecture successfully caters to this requirement. An efficient functionality based framework classification employing logical interfaces among different frameworks reduces the time-to-market, making it even more feasible for product deployment. A proof-of-concept extending android platform was illustrated to evaluate the feasibility of this architecture. A related aim of our paper is to inform the scientific community about the design in new environments by providing Execution and Meta architecture views of such a system. Finally, we concluded by presenting MSC (Message State Chart) used for trial experiment

conducted. Initial results reveal the said architecture design acts as the central building block for android to cater IMS based services.

Future work includes support for upgrade/downgrade among various IMS features. E.g., to upgrade a VOIP call to a VVOIP call dynamically.

ACKNOWLEDGEMENTS

The authors would like to thank Aricent Communications for fostering and encouraging innovations on handheld devices, especially on Android.

REFERENCES

[1] L Roach, A., "SIP-Specific Event Notification," IETF RFC 3265, June 2002.

[2] Rosenberg, J., Schulzrinne,H., et. al. "SIP:Session Initiation Protocol", RFC 3261, June 2002.

[3] S. Tsang, D. Marples, S. Moyer, "Accessing networked appliances using the session initiation protocol", ICC 2001 - IEEE International Conference on Communications, no. 1, June 2001 pp. 1280-1285.

[4] Baldus, H., Baumeister, M., Eggenhuissen, H., 2000, Montvay, A., Stut, W., WWICE: An Architecture for In-Home Digital Networks. Proc. SPIE, Vol. 3969, 196-203.

[5] Enck, W.; Ongtang, M.; McDaniel, P., "Understanding Android Security", IEEE Security & Privacy, 7(1), pp.50-57, 2009.

[6] A. I. Wang, C.-F. Sørensen, and T. Fossum. Mobile Peer-to-Peer Technology used to Promote Spontaneous Collaboration. In The 2005 International Symposium on Collaborative Technologies and Systems (CTS 2005), page 8, Saint Louis, Missouri, USA, May 15-19 2005.

[7] Stan Moyer, Dave Marples, Simon Tsang, "A protocol for wide-area secure networked appliance communication", IEEE Communications Magazine, vol. 39, no. 10, October 2001 pp. 52-59.

[8] A.W. Brown, J. Conallen, and D. Tropeano, Introduction: Models, Modeling, and Model-Driven Architecture(MDA), in Model Driven Software Development, Volume 2, Research and Practice in Software Engineering, S. Beydeda, M. Book, and V. Gruhn, Eds. New York, Springer-Verlag, 2005, pp.1-16.

[9] V. Marques, R. Aguiar, C. Garcia, J. Moreno, C. Beaujean,E. Melin, and M. Liebsch. IP-based QoS architecture for 4G operator scenarios. IEEE Wireless Communications Magazine, June 03.

[10] Mobydick: Mobility and differentiated services in a future IP network – Final report," http://www.ist-mobydick.org, April 2004.

[11] A.W. Brown, J. Conallen, and D. Tropeano, Introduction: Models, Modeling, and Model-Driven Architecture(MDA), in Model Driven Software Development, Volume 2, Research and Practice in Software Engineering, S. Beydeda, M. Book, and V. Gruhn, Eds.New York, Springer-Verlag, 2005, pp.1-16.

[12] D. Waddington and P. Lardieri, "Model-Centric Software Development," IEEE Computer, vol. 39, No. 2, pp. 28-30, February 2006.

[13] Stan Moyer, Amjad Umar, "The impact of network convergence on telecommunications software", IEEE Communications Magazine, vol. 39,no. 1, January 2001 pp. 78-84.

[14] Video Quality Measurement Techniques, Stephen Wolf, Margaret Pinson, NTIA Report 02-392, June 2002.

[15] Objective video quality assessment system based on human perception AA Webster, CT Jones, MH Pinson, SD Voran, S Wolf Proc. of SPIE, 2003.

Authors

Mr. Suman Kumar is a part of Research Group in Aricent Communications. He is involved in building proof-of-concepts from design to execution focusing primarily on NEXT GEN Architectures for Mobile handheld devices. He has worked on several international assignments in companies like Microsoft, Redmond (USA) from architecture to execution of state-of-art wireless technologies for handheld devices. He and his engineering team drive various technological roadmap activities, engages with academia, file patents and publish papers in competitive conferences and journals for the research community.

Mr. Vijay Anand is Director - Engineering in Mobile Terminal Products Division of ARICENT Technologies Limited. In this capacity, he directs research activities that drive the state-of-art in the design of next-generation wireless/Multimedia technologies, focusing primarily on NEXT GEN Design architecture for Mobile handheld devices. He has 2 software patents to his credit and has chaired several technical sessions. He has given tutorials on "Mobile Device Architecture: Present and Future", "Digital Living Network Alliance", "Fixed Mobile Convergence" at various International Conferences and symposia like NCC 10, NTMS 07, WPMC 07, CSI 07, IEEE Wi-MAX, ICEMC2 , AICT & MAP. He is a sought after panelist, keynote speaker, and architect on next generation mobile terminals.