Anexo I

Don Agustín Trujillo Pino, SECRETARIO DEL DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS DE LA UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA,

CERTIFICA,

Que el Consejo de Doctores del Departamento, en su sesión extraordinaria de fecha 12 de Noviembre de 2015, tomó el acuerdo de dar el consentimiento para su tramitación a la tesis doctoral titulada "***Control de interrupciones de video streaming móvil en arquitecturas Android usando técnicas de realidad aumentada y WebRTC***" presentada por el doctorando Don Diego Marcillo Parra y dirigida por el Doctor Don Álvaro Suárez Sarmiento y la Doctora Doña Elsa María Macías López.

Y para que así conste, y a efectos de lo previsto en el Artº 6 del Reglamento para la elaboración, defensa, tribunal y evaluación de tesis doctorales de la Universidad de Las Palmas de Gran Canaria, firmo la presente en Las Palmas de Gran Canaria, a Doce de Noviembre de Dos Mil Quince.

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Departamento de Informática y Sistemas

Doctorado en
Tecnologías de la Información y sus Aplicaciones

Tesis Doctoral

Control de interrupciones de video streaming móvil en arquitecturas Android usando técnicas de realidad aumentada y WebRTC

Autor

Diego Marcillo Parra

Director(s)

Dr. Álvaro Suarez Sarmiento

Dra. Elsa Mª Macías López

El (Los) director(es)

El doctorando

Las Palmas de Gran Canaria, 30 octubre de 2015

Agradecimientos

Agradezco a mi Dios Padre Celestial por su fortaleza y guía para cumplir con este gran reto y sueño profesional. A mi preciosa esposa Elena, a mis adorados hijos Camila y Mateo, por el tiempo que me han concedido, un tiempo en el que no he podido ser parte de la historia familiar, pero que siempre han sido mi fuente de amor, paciencia, inspiración, fortaleza y alegría, este trabajo es también el suyo.

A mi querida familia, a mis padres y hermanos, por haber sido el apoyo y fuerza en cada momento, en cada desánimo para seguir en este duro camino de mi vida. A Silvana e Iván por ser mis amigos incondicionales en este gran sueño y a mis amigos de siempre, por sus palabras de aliento y motivación diaria que me han permitido seguir adelante en este reto.

Con todo mi corazón, un sentido gracias a Elsa y Álvaro, por su paciencia, apoyo y palabras de ánimo que apuntalaron a levantarme de mis caídas y seguir perseverando en esta hermosa meta de mi vida.

A ti Tatiana, amiga y hermana, compañera de esta travesía, que con tu gran apoyo profesional, moral y humano, me han permitido continuar en los momentos difíciles de este trabajo y esta profesión.

Resumen

Los avances en la Tecnología de la Información y la Comunicación han impactado la Sociedad muy sorprendentemente, constituyéndose en un elemento de vital importancia. El ser humano está en una constante búsqueda de nuevas tecnologías, que simplifiquen las actividades en cualquiera de las áreas que necesite. En la Comunicación inalámbrica se ha incrementado el ancho de banda usando nuevas técnicas de codificación, se ha mejorado mucho la arquitectura hardware de los terminales móviles y sus sistemas operativos. El hardware puede incluir hasta ocho núcleos, pantallas flexibles que faciliten el consumo de contenidos de realidad virtual y de realidad aumentada. Los sensores permiten diseñar aplicaciones inteligentes e interactivas con el medio físico. Los sistemas operativos optimizan el consumo energético y manejo eficiente de datos multimedia (audio y vídeo). Servicios móviles como la videoconferencia y streaming de video almacenado son muy habituales a día de hoy; por lo que es imperativo garantizar la prestación del servicio con criterios de calidad de experiencia. Desafío complejo porque la gran cantidad de tráfico que demanda este servicio es difícil de procesar en los limitados recursos de los terminales móviles. Pero más complejo es el hecho que el canal inalámbrico tiene un comportamiento impredecible ocasionado por la desconexión impredecible del canal inalámbrico que causa interrupciones del servicio de video. Esto afecta muy negativamente a la Calidad de la experiencia del usuario y pone en peligro el futuro éxito de este servicio. Por eso, en esta tesis se ha contribuido a mitigar este problema, contribuyendo con varias soluciones software que se aplican en diferentes ambientes. Para ello se ha modelado con patrones de diseño una arquitectura básica de un esquema general de video streaming. Para entender el problema que afecta al servicio de video streaming, se modeló y generó un modelo matemático que permite demostrar claramente que el rendimiento del servicio ante interrupciones es muy pobre.

La primera solución consiste en utilizar proxies para mitigar los efectos adversos de una interrupción de video streaming; permitiendo al cliente continuar consumiendo el contenido multimedia desde la posición en la que se encontraba, dado el establecimiento del canal de comunicación. Posteriormente se planteó un modelo basado en realidad aumentada, donde el usuario puede observar en tiempo real mediante técnicas de realidad aumenta, como se encuentra el estado del canal de comunicación en un lugar dado; para

con esta información tome acciones que permitan la continuidad del contenido multimedia, evitando entrar en zonas de interrupción. Se utilizó el modelo base en forma iterativa con incrementales de funcionalidad mediante el uso de patrones software y reglas de negocio que evalúan el estado del canal de comunicación a través del sensado del dispositivo móvil, entregando información que permite desplegar consultas colaborativas debido a que arma mapas online con la información obtenida sobre la ruta seguida por el usuario mediante archivos KML. Estos archivos pueden ser consumidos por herramientas como Google Earth y Google Maps

Finalmente, se planteó un modelo para video en tiempo real que permita controlar la interrupción de una sesión de videoconferencia por video streaming, debido a una disrupción del canal de comunicación inalámbrico. Se mantuvo el modelo base en forma iterativa con incrementales en sus dos subsistemas: establecimiento y externo. Cada uno de estos subsistemas se desarrolló con base en patrones software de diseño y funcionalidades adicionales para cumplir con el objetivo de proveer control de interrupción y continuidad del servicio de videoconferencia.

Ninguna de las soluciones anteriores supone un coste adicional en el tiempo de ejecución y se ha definido un mecanismo que permite reestablecer la sesión; continuar con la visualización del contenido multimedia desde la posición en la que se encontraba, cuando se produjo la disrupción. Minimizando significativamente el impacto negativo de la pérdida de sesión y mejorando la experiencia de usuario en servicios de videoconferencia y video streaming.

Palabras clave

Video streaming, comunicación inalámbrica, dispositivos inalámbricos, agentes inteligentes, servicios Web, disrupción, interrupción, recuperación, video streaming bajo demanda, video streaming en tiempo real, WebRTC, HTML5, DASH, sensores, reglas de negocio, patrones software de diseño, realidad aumenta, KML, KMZ.

Abstract

Advances in information technology and communication have impacted surprisingly in the society, being an element of vital importance. Human beings are in constant search for new technologies that reduce the activities in any of the needed areas. In wireless communication the bandwidth has been increased using new coding techniques, the hardware architecture of mobile devices and their operating systems has been greatly improved. The hardware can include up to eight cores, flexible displays to facilitate the use of virtual and augmented reality. Sensors allow to design smart applications with the physical environment. Operating systems optimize energy consumption and efficient management of multimedia data (audio and video). Mobile services such as videoconferences and video streaming are very common these days, so it is imperative to ensure the service with quality criteria of experience. This is a complex challenge because of the amount of traffic that this service demands, is difficult to process in the limited resources of mobile terminals.

The wireless channel has an unpredictable behavior caused by the unpredictable wireless channel disconnection that causes interruptions in the video service. This negatively affects to the quality of user experience and puts in danger the future success of this service. Therefore, this thesis has contributed to mitigate this problem with several software solutions that apply in different environments.

For this reason a basic architecture of a general scheme of video streaming has been modeled with design patterns. To understand the problem that affects the video streaming service, it was modeled and created a mathematical model to demonstrate clearly that service performance facing interruptions is very poor.

The first solution is to use proxies to mitigate adverse effects of a video streaming interruption; allowing the costumer continue consuming the multimedia content from the position he was when the channel communication was stablished.

After that a model based on augmented reality was raised, where the user can observe in real time trough augmented techniques how is the status of the communication channel in a specific place, for this information I took decisions that

allow the continuity of multimedia content, avoiding entering in interruption areas. The base model was used interactively with increments of functionality through the use of software patterns and business rules that evaluate the status of the communication channel by sensing the mobile device, providing information that allows to deploy collaborative consultations because online maps are formed with the obtained information of the route followed for the user through KML files. This files can be consumed by tools like Google Earth and Google Maps.

Finally, a model for video in real time that allows to control the interruption of a videoconference by video streaming was raised, due to a disruption of the wireless communication channel. The base model was maintained in an iteratively way with incremental in two systems: establishment and external. Each of these subsystems was developed based on design software patterns and additional features to get the goal of provide interruption control and continuity of videoconference service.

None of the previous solutions has an additional cost in the executing time and a mechanism to reestablish the session, continue displaying the media from the position that the costumer was when the disruption occurred was defined. Significantly minimizing the negative impact of the loss of session and improving the user experience in videoconferences and video streaming.

Keywords

Video streaming, wireless communication, wireless devices, smart agents, web services, disruption, interruption, video streaming on demand, video streaming in real time, WebRTC, HTML5, DASH, sensors, business rules, design software patterns, augmented reality, KML, KMZ.

Índices

Índice de Contenido

Índice de Tablas

Índice de Figuras

1. Introducción

En este capítulo se presenta: las motivaciones y contexto del problema que ha sugerido la realización de este trabajo de investigación, los objetivos y aportaciones de esta tesis doctoral y por último presentamos la estructura de esta memoria.

1.1. Motivaciones

Con la penetrante aceptación de las redes inalámbricas sobre dispositivos inalámbricos en ámbitos empresariales, residenciales y públicos, mediante el uso de los estándares como: *Wireless Fidelity* (*WiFi*) [1]; *Worldwide Interoperability for Microwave Access* (*WiMAX*) [2], *3rd Generation Partnership Project* (*3GPP-3GPP2*) [2], [3]; *Universal Mobile Telecommunications System* (*UMTS*) [4] y *Long Term Evolution* (*LTE*) [5], han permitido visualizar contenidos multimedia en tiempo real mediante la utilización de tecnología de *streaming*.

Video streaming es una técnica que consiste en objetos multimedia que se reproducen y descargan simultáneamente desde el Internet sin la necesidad de almacenar toda la información en la memoria del cliente, por tanto, no requiere que el video se almacene en el teléfono móvil. El problema para implantar la visualización de contenidos multimedia en dispositivos inalámbricos en tiempo real o bajo demanda, radica en varios aspectos: limitaciones de procesamiento, memoria, consumo de batería, pantalla de visualización y un comportamiento caótico espacio-temporal e irregular de los canales de comunicación inalámbricos (todas las tecnologías sufren de este inconveniente, en especial WiFi que provoca impredecibles pérdidas de cobertura), que provocan desconexiones radio por varias causas: deterioro de la señal, fuera de cobertura, entre otros [6]. Estas impredecibles desconexiones afectan la transmisión del contenido multimedia mediante tecnología video streaming, haciendo que sea necesario el inicio de una nueva sesión o siendo muy pesimista el abandono de la misma. Los buffers de visualización mitigan desconexiones de muy corta duración. Si la desconexión es de larga duración, el buffer se vacía y se corta la visualización. Es imperioso, plantear una solución al problema que afecta a la transmisión de contenido multimedia mediante mecanismos, que permitan controlar el restablecimiento de la comunicación y posterior visualización del contenido multimedia desde la posición en la que se encontraba, el cliente, al momento de ocurrir la interrupción.

Una de las primeras aproximaciones realizadas por el grupo de investigación, establece un mecanismo basado en proxies, usado en computadores portátiles y dispositivos inalámbricos basados en sistema operativo Symbian [7]. Este mecanismo

implementa el uso de agentes para recuperar sesiones *Real Time Streaming Protocol* (*RTSP*) [8], desarrollados en *Java Agent Development Framework* (*JADE*) [9] para portátiles y JADE *Lightweight Extensible Agent Platform* (*LEAP*) [10] para dispositivos inalámbricos.

Un *Proxy* es un conocido patrón de diseño software. Por ello, la principal motivación de esta tesis doctoral es lograr que esta arquitectura puede ser usada como un modelo de base para conseguir soluciones aplicables a varios tipos de dispositivos inalámbricos utilizando patrones de diseño software más potentes.

1.2. Contexto del problema

El constante desarrollo tecnológico de los dispositivos inalámbricos y las redes de comunicación inalámbrica han despertado mucho interés en los usuarios por el consumo de aplicaciones para visualizar contenido multimedia en tiempo real o bajo demanda. Los teléfonos móviles hoy en día, son dispositivos poderosos con varias interfaces de redes inalámbricas heterogéneas, que permiten recibir la información multimedia a través de la tecnología video streaming a bajo costo. Video streaming es muy apropiada para aplicaciones cliente que corren en dispositivos portables (celulares, agenda personal de bolsillo, reproductor de audio digital) debido a su limitada memoria y ancho de banda; esta técnica trata de ocultar la latencia de la red en una sesión entre el cliente y el servidor, consiguiendo que el tiempo de espera por tramas de video consecutivas sean mínimas. Sin embargo, las interrupciones del servicio de video streaming o videoconferencia provocan que el servidor continúe enviando información generando gasto de recursos del servidor y sobrecarga en el canal. Mitigar estos efectos es importante porque con video streaming es posible acceder a contenidos de televisión digital que proporcionan cadenas de televisión, esto permite el aumento de sus cuotas de audiencia a un precio razonable.

Mitigar las interrupciones de servicio en video streaming móvil en Internet con redes de acceso móvil e inalámbricas es un problema que aún no ha sido solucionado eficientemente, existe dificultad de controlar las interrupciones intermitentes

ocasionados por pérdida de cobertura u otras causas en redes inalámbricas. La pérdida de conexión puede ser por naturaleza muy variada y producirse a diferentes niveles de la arquitectura de red; surgen entonces dificultades de recepción de datos multimedia, cierre de la sesión actual y obligatoriedad de abrir una nueva sesión, gastos de los recursos del servidor al enviar tramas multimedia desconociendo el estado real del cliente; todo esto confluye a la perdida de efectividad de video streaming y a la ineficiencia de RTSP, ocasionando también pérdida de tiempo del usuario que abandona la sesión de forma definitiva.

Un problema importante que las redes inalámbricas presentan ante un servicio de video streaming es la baja fiabilidad a fluctuaciones de ancho de banda que conducen a la degradación de la calidad de video de manera significativa [11][12]. Especialmente en servicios de video streaming multimedia móvil donde las prestaciones son muy altas de ancho de banda y velocidad de transmisión de datos, para satisfacer la demanda del Mercado. Los operadores móviles están aprovechando WiFi para aliviar la presión de la demanda de ancho de banda creciente de aplicaciones y están tratando de proveer mecanismos inteligentes que controlen la forma que acceden los usuarios a las redes WiFi. Esta falta de controles degrada la Calidad de la Experiencia de usuario (QoE, del inglés *Quality of Experience*). Es importante el aprovisionamiento de la Calidad de Servicio (*QoS*, del inglés *Quality of Service*) de extremo a extremo, necesario para el mantenimiento continuo de reproducción de video en las aplicaciones multimedia en tiempo real. Condición que se ve afectada por la realidad del canal inalámbrico que es muy variable en el tiempo, debido a interferencia, fading, y movilidad; provocando que el servicio de video streaming media no pueda ser difundido con calidad [13][14].

Múltiples factores afectan a las redes inalámbricas al realizar video streaming, problemas como ancho de banda variable en el tiempo, efectos de atenuación, degradación de señal, interferencias, retraso, jitter o alta tasa de pérdida de paquetes. Si el remitente transmite más rápido que el ancho de banda disponible, entonces la congestión ocurre. Los paquetes se pierden y hay una severa caída de la calidad de video. Si el emisor transmite más lento que el ancho de banda disponible entonces el receptor produce una calidad de video óptima [15] [16] [17].

Sin embargo, la transmisión de datos en los canales inalámbricos sufren de muchos errores, pérdidas frecuentes de paquetes y la cobertura de radio no siempre es constante, por lo que se produce interrupciones frecuentes del teléfono móvil. Estas interrupciones son totalmente impredecibles y únicamente se puede mitigar su efecto adverso sobre la comunicación [18]. La transmisión del video streaming en redes inalámbricas enfrenta desafíos de las condiciones del canal y recursos de red limitados. Por lo tanto, para un eficiente video streaming se requiere de una eficiente QoS y un mecanismo de control. Esto permite mitigar la inestabilidad de las redes inalámbricas a problemas como ancho de banda variable en el tiempo y limitado, y la congestión de tráfico durante la transmisión de una ráfaga de flujos [19] [20].

El servicio de video streaming presenta un proceso complejo de percepción por el usuario en su sistema visual y auditivo, combinado con el sistema sensorial y cognitivo. La relación que presentan los aspectos sensorial y auditivo con el video determinan aspectos como: brillo, color, forma, movimiento, tono de audio, volumen y timbre [21][22]. La QoE para los servicios multimedia se relaciona fuertemente con la QoS, rendimiento del terminal y atributo de servicio. La QoE es una medida del rendimiento de los niveles de servicio video streaming bajo la perspectiva del usuario, proporciona una medida subjetiva que cuantifica el impacto que tiene en el usuario la presencia de fallos en el servicio. Estos fallos pueden ser determinados por métricas de QoE como: la duración de los fallos en el servicio, errores por segundo y segundos sin disponibilidad del servicio [23]. En relación al rendimiento de un dispositivo móvil ante un video streaming, tenemos factores que afectan drásticamente la QoE por limitaciones de capacidad del procesamiento, capacidad de visualización, velocidad de procesamiento, capacidad de almacenamiento y tamaño del buffer de memoria, factor clave para mitigar el retraso y el jitter [24].

La Unión Internacional de Telecomunicaciones (ITU-T, del inglés *International Telecommunication Union*) desarrolló modelos de QoE estandarizados para predecir la calidad de audio, video y voz; utilizando de forma subjetiva *Mean Opinion Score* (*MOS*) [24] [25] [26]. En [27] se analiza los enfoques de medición de calidad de video propuestos para evaluar la diferencia de calidad entre fotografías *Peak-Signal-to-Noise-Ratio* (*PSNR*), para medir los efectos de percepción de las deficiencias de video *Video*

Quality Metric (*VQM*), métricas de la calidad de imagen en movimiento que incorpora dos características de visión humana: sensibilidad al contraste y enmascaramiento *Moving Pictures Quality Metric* (*MPQM*), evalúa la calidad de video basado en la medición de distorsión estructural ya que la visión humana es altamente especializada en extraer información estructural *Structural Similarity Index* (*SSIM*), métricas como: medida de distorsión, calidad del ruido, variación en relación a la sensibilidad con la distancia, dimensiones de la imagen *Noise Quality Measure* (*NQM*) [28] [29] [30].

1.3. Objetivos de la tesis

El objetivo principal de esta tesis doctoral es demostrar que el modelo base apoyado en el patrón software de diseño *Proxy* puede ser re-utilizado para conseguir soluciones de mitigación de los efectos adversos del streaming móvil, más potentes.

Como primer objetivo planteamos el diseño novedoso de un modelo basado en patrones diseño de software, para el control de interrupciones en sesiones de video streaming, en dispositivos inalámbricos con sistema operativo Android y su desarrollo utilizando el framework JADE mediante el add-on JADE-Android [31], [32]. Nuestro interés es definir el modelo base mediante el uso del patrón *Modelo Vista Controlador* (*MVC*) [33], con el objetivo de reducir el esfuerzo de programación y garantizando la actualización y mantenimiento del software de forma sencilla y en reducido espacio de tiempo. Esta implementación en niveles y por separado de cada elemento que conforman una aplicación en el MVC, permite lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores. Esto permite analizar a fondo los componentes que conforman el modelo y sus interrelaciones, para determinar la validez y eficacia del modelo ante una interrupción de la sesión de video streaming. Fundamentado con el patrón Modelo Vista Controlador, permite controlar la interrupción de la sesión de video streaming debido a una disrupción del canal de comunicación inalámbrico, garantizando la continuidad del servicio en dispositivos inalámbricos que utilicen sistema operativos Android e influyendo positivamente en la QoE del usuario al optimizar el tiempo de ejecución del video streaming, eliminar las retransmisiones y minimizar la congestión del canal de comunicación inalámbrico.

El segundo objetivo es generar una solución base que utilice el mecanismo de control de interrupciones para sistema operativo Android, que permita generar un novedoso aplicativo que controle las interrupciones y elimine los reinicios de sesión de video streaming, permitiendo recibir los flujos de video almacenados temporalmente en un buffer cuando el dispositivo móvil recupere conexión una vez que haya ingresado a zona de cobertura.

El tercer objetivo es incrementar la funcionalidad al modelo base, mediante patrones software de diseño que utilicen sensado del dispositivo móvil con sistema operativo Android, para lograr que el mecanismo de control de interrupciones, brinde un modelo con realidad aumentada que permita alertar y actuar proactivamente al usuario ante un posible ingreso a zona de interrupción, mediante la publicación en pantalla de mensajes de alerta e información del nivel de intensidad de señal y la consulta de información de mapas online de rutas de cobertura; con el propósito de recudir el impacto negativo que ocasiona una interrupción y permitiendo la continuidad del servicio.

El cuarto objetivo es acoplar el modelo base e incrementar su funcionalidad mediante patrones software de diseño, para controlar la interrupción del video en tiempo real de la sesión del servicio de videoconferencia basado en navegador Web, garantizando la continuidad del servicio en cualquier dispositivo móvil o computador, elevando el positivamente el nivel de calidad de experiencia de usuario.

1.4. Contribuciones de la tesis

Para llevar a cabo los objetivos de la tesis se han realizado las siguientes contribuciones:

Diseño del modelo base fundamentado en patrones software: el diseño contiene dos subsistemas: base y externo. El subsistema base conformado por la implementación del patrón MVC, para facilitar el mantenimiento, la escalabilidad y la reutilización de la aplicación. En cada una de las capas del patrón MVC se implementa patrones software adicionales con el objetivo de fortalecer el modelo: en la capa vista se utiliza el patrón *Composite*; en el Controlador se aplica los patrones software *Proxy* y *Strategy*; en el

Modelo se utiliza el patrón *Observer*. El subsistema externo implementa el patrón *Observer*. El *Proxy* del subsistema base se encarga de implantar la comunicación entre subsistemas, permitiendo así al *Observer* del subsistema externo reportar el estado del canal de comunicación al *Proxy* para determinar si existe una interrupción. Este diseño permitió establecer el mecanismo de control de interrupciones de sesiones de video streaming, por desconexiones del canal de comunicación inalámbrico en dispositivos inalámbricos con sistema operativo Android.

Desarrollo de la plataforma de control de interrupciones en ambientes Android: utilizando el diseño del modelo base, se desarrolló el mecanismo de control de interrupción. Este mecanismo utiliza el patrón *Proxy*, para recibir del video player la solicitud del servicio y pide al servidor video streaming el envío del video al cliente, mediante el uso de video streaming se procede a la entrega de flujos al video player. El patrón observador del modelo está evaluando constantemente el estado del almacenamiento temporal, si el buffer supera el umbral comunica al *Proxy*, para que éste solicite el servicio del patrón observador del subsistema externo acerca del estado del canal de comunicación. Si el estado que entrega el *Observer* al *Proxy* es desconectado el canal, *Proxy* solicita a *Strategy*, la ejecución del algoritmo de interrupción. El algoritmo de interrupción realiza dos actividades: solicitud del almacenamiento temporal de flujos emitidos por el servidor video streaming y despliegue en la vista del mensaje *Reconectando*, mientras dure la desconexión. Una vez superada la desconexión, el *Observer* del subsistema externo determina que el canal de comunicación se encuentra disponible e informa al *Proxy*, quién solicita a *Strategy* la ejecución del algoritmo de conexión; generando la transmisión al cliente de los flujos almacenados en el buffer desde la posición en la que se encontraba el usuario al ocurrir la interrupción. Con la experimentación práctica hemos demostrado que el modelo base permite reducir el tiempo de ejecución de una sesión de video streaming ante varias interrupciones en el canal inalámbrico.

Diseño del modelo con realidad aumentada: el diseño mantiene el subsistema base del modelo base e incrementa su funcionalidad en el subsistema externo con el uso de los patrones software *Observer* y *Adapter*. En la capa modelo se implementa un patrón de diseño *Observer* que está monitoreando al canal de comunicación y de acuerdo a su

estado implementa la regla de negocio correspondiente para desplegar en la vista la información al usuario con realidad aumentada sobre la intensidad de la señal y su geolocalización, y construir el archivo de datos geográficos, que incluye información de geolocalización; *Received Signal Strength Indicator* (*RSSI*) y captura de la pantalla del video en ese instante, y mediante el uso del patrón *Adapter* generar el archivo correspondiente a ser consumido por la aplicación Google Earth. En la capa vista, que se despliega en los dispositivos inalámbricos, se utiliza el *Composite* para entregar una interfaz de usuario en la se adiciona los siguientes componentes: parámetros dispositivo, *Global Positioning system* (*GPS*), *WiFi*, *grabar video*; *datos con realidad aumentada*, *georeferencia*, *nivel RSSI*. En la capa controladora se incrementa al patrón *Observer* externo del canal de comunicación dos listener para RSSI y GPS. El *Proxy* del subsistema base mantiene la función de estar encargado de implantar la comunicación entre subsistemas, permitiendo así al observador de la capa controladora del subsistema externo reportar el estado del canal de comunicación al *Proxy* para determinar si existe una interrupción. Las nuevas funcionalidades incrementadas a este diseño permiten utilizar sensado del dispositivo móvil con sistema operativo Android, para lograr que el mecanismo de control de interrupciones, brinde un modelo de experiencia inmersiva del servicio de video streaming al usuario, mediante despliegue de información con técnicas de realidad aumentada.

Desarrollo de la plataforma con realidad aumentada: utilizando el diseño del modelo con realidad aumentada, se desarrolló el mecanismo de control de interrupción. Este mecanismo inicia cuando el video Player solicita el despliegue del video streaming al *Proxy*, éste a su vez lo instancia del servidor video streaming. De igual manera en forma continua se registra el estado de los recursos del dispositivo: nivel de RSSI, información del video y geolocalización. El estado de estos recursos se revisa por el patrón observador externo del subsistema externo y comunicados al *Proxy*. Para la publicación de esta información, en la vista utilizamos la interfaz de datos con realidad aumentada y en el modelo se genera el archivo de datos geográficos para ser consumido por la aplicación Google Earth. *Proxy* solicita al *Observer* externo canal del subsistema externo el estado del nivel de RSSI y geolocalización para entregarlos al *Observer* externo del modelo para el análisis y determinar una posible interrupción e informar al *Proxy* del

subsistema base. Dado este evento *Proxy* solicita al *Strategy* la ejecución del algoritmo de interrupción, el cual demanda que se almacene en almacenamiento temporal los flujos del video que no pueden verse en video player, despliega en la vista el mensaje de que esta desconectado e inicia el bucle de reconexión. Al recibir la notificación de canal restablecido, *Proxy* solicita al patrón *Strategy* la ejecución del algoritmo de conexión, el cual permite desplegar el video desde el instante que hubo la interrupción, permitiendo la continuidad del video al cliente. Con la experimentación práctica se ha demostrado que es factible iterar el modelo base e incrementar nuevas funcionalidades. La ejecución de estas nuevas funcionalidades se solapa con el tiempo de visualización del video, por lo que no existe un coste adicional de tiempo. Al ocurrir interrupciones en la transmisión del video streaming, el control proactivo que realiza ésta plataforma apoya en la reducción del tiempo de ejecución del video.

Diseño del modelo para video en tiempo real con control de interrupciones: el diseño se encuentra definido por tres subsistemas: establecimiento; sesión y externo, y tres patrones software de diseño: MVC, *Observer* y *Adapter*. El MVC ha sido seleccionado con miras a lograr independizar las capas de desarrollo, mejorar la gestión de cambio y escalabilidad. El patrón de diseño *Observer* es el encargado de analizar el estado del objeto canal de comunicación. El *Adapter* es el encargado de convertir el audio y video almacenado temporalmente en el buffer del servidor de aplicaciones, y una vez convertido almacenarlo temporalmente en cada cliente peer y su posterior despliegue. El subsistema establecimiento, contempla tres elementos: Modelo; Vista; Controlador. El Modelo se encuentra ubicado en el servidor de aplicaciones, encargado de proveer almacenamiento temporal a ser utilizado para recibir de los clientes Web, el buffer almacenado dada una interrupción del canal y almacenamiento de la base de datos de usuarios. Este Modelo adiciona y se fortalece con la implementación del *Adapter*, que convierte los datos de audio y video almacenados en flujos para ser enviados a los clientes Web opuestos. La vista se despliega en los dispositivos inalámbricos utilizando el *Composite*. Es una interfaz de usuario que presenta los formularios para: permitir el acceso a los recursos de la cámara y micrófono de cada dispositivo móvil; creación de usuario de videoconferencia e inicio de videoconferencia. El Controlador se ubica en el servidor de aplicaciones, está conformado por el *Proxy*.

Éste se encarga de validar la existencia de usuario y de solicitar el establecimiento de la sesión Web para videoconferencia. El subsistema sesión, presenta tres elementos: Modelo, Vista y Controlador. El Modelo se encuentra ubicado en cada cliente Web, encargado de proveer almacenamiento temporal para guardar los datos de audio y video capturados por la cámara y micrófono del cliente, si presenta una interrupción el canal. Este Modelo se fortalece con la implementación de un *Observer* que constantemente está monitoreando el estado del buffer. Cuando el buffer supera el umbral establecido, se encarga de notificar al *Proxy* de este evento. La vista se despliega en los dispositivos inalámbricos utilizando el *Composite*. Es una interfaz de usuario que presenta los formularios para: finalizar video conferencia y salir plataforma; además del despliegue de la videoconferencia y de las estadísticas de red del servicio de videoconferencia. En caso de ocurrir una interrupción, despliega un mensaje *Reconectando*, el cual permanece activo hasta cuando el canal se restablezca y la videoconferencia continúe y se adicione el despliegue del video almacenado en el buffer por el usuario peer opuesto. El Controlador se ubica en el servidor de aplicaciones, este está conformado por: *Proxy* y *Strategy*. El *Proxy* valora la estrategia a seleccionar en función del estado que le entrega el *Observer* del modelo y el *Observer* externo canal del subsistema externo. En el subsistema externo se usa un *Observer* que permite validar el estado del canal de comunicación ante una alerta del *Observer* del almacenamiento temporal de la capa modelo del subsistema base. Las nuevas funcionalidades a este diseño permiten controlar la interrupción de una sesión de Web videoconferencia y video streaming de video, debido a una disrupción del canal de comunicación inalámbrico. Se mantuvo el modelo base en forma iterativa con incrementales en sus dos subsistemas: establecimiento y externo. Cada uno de estos subsistemas se desarrolló con base en patrones software de diseño y funcionalidades adicionales para cumplir con el objetivo de proveer control de interrupción y continuidad del servicio de Web videoconferencia y video streaming de video.

Desarrollo de la plataforma para video en tiempo real con control de interrupciones: utilizando el diseño del modelo para video en tiempo real con control de interrupciones, se desarrolló la arquitectura donde el cliente *Web Real-Time Communication* (*WebRTC*) realiza la solicitud de servicio de videoconferencia al *Proxy* del subsistema

establecimiento, éste invoca al servidor WebRTC quien solicita permiso al cliente WebRTC para acceder al recurso de la cámara y micrófono del dispositivo móvil y la creación de un usuario. *Proxy* al recibir el usuario creado invoca al algoritmo validación para verificar si es nuevo o existe en la base de datos de usuario. Si es nuevo lo crea y registra en la base de datos, si existe solicita a la base de datos la información del usuario para entregarle al servidor WebRTC. Creados los usuarios para los clientes WebRTC, *Proxy* espera que uno de ellos solicite el inicio de videoconferencia para pedir a estrategia el algoritmo establecimiento. Una vez establecida la sesión, el *Proxy* del subsistema sesión activa el *Observer* de Modelo para que informe si existe algún cambio en el umbral del almacenamiento temporal. Dada la alerta por *Observer*, el *Proxy* solicita confirmación mediante el monitoreo del *Observer* externo canal del subsistema externo sobre el estado del canal de comunicación. Con un estado de interrupción, el *Proxy* solicita a estrategia implantar el algoritmo interrupción que ejecuta dos actividades: solicitar el almacenamiento temporal de los flujos de audio y video emitidos por los clientes WebRTC en los dispositivos inalámbricos y comunicar a sus vistas de esta situación mediante un mensaje *Reconectando*. Si el estado del canal indica que está operativo, *Proxy* solicita a estrategia cambiar al algoritmo conexión para solicitar el restablecimiento de la sesión de videoconferencia, por lo que es llamado nuevamente el subsistema establecimiento. Una vez establecida la sesión, el *Proxy* solicita al almacenamiento temporal de los clientes enviar este buffer almacenado al almacenamiento del servidor de aplicaciones, para en éste llamar al patrón *Adapter* para que convierta el flujo de audio y video y sea enviado nuevamente a los clientes WebRTC opuestos para despliegue en vista de cada dispositivo móvil. Con la experimentación práctica se ha demostrado que es factible mantener el modelo base e incrementar o reemplazar nuevas funcionalidades, existe un incremento de coste por el tiempo de establecimiento de la sesión de videoconferencia que es mínima y no afecta al tiempo de ejecución de la videoconferencia y video streaming de video ante varias interrupciones del canal de comunicación inalámbrica.

En cuanto a publicaciones, este trabajo ha dado lugar a la publicación:

- *T. Gualotuña, D. Marcillo, E. M. López, and A. Suárez-Sarmiento*, **Mobile Video Service Disruptions Control in Android Using JADE**" in *Advances in Computing*

and Communications, Springer, 2011, pp. 481–490*,* que presenta la arquitectura basado en proxies y agentes inteligentes para el control de interrupciones en ambientes inalámbricos en dispositivos Android, que resuelve la interrupción y reanuda automáticamente la sesión de video streaming sin pérdida de la información mediante la aplicación de almacenamiento temporal intermedio.

En relación a la definición y ejecución de proyectos de investigación como Director de proyecto se ha dado lugar a:

- ***Prototipo de un sistema ubicuo de localización y navegación autónoma usando dispositivos móviles***, que promueve el desarrollo de aplicaciones que permiten utilizar realidad aumentada y geolocalización para ubicación de lugares en la Escuela Politécnica del Ejército (ESPE), Quito, Ecuador.
- ***Diseño de un mecanismo para control de interrupciones en servicios de video streaming en dispositivos Android***, que permitió el desarrollo del aplicativo para dispositivos Android para la recuperación de sesiones de video streaming frente a una interrupción.
- ***Sistema de monitorización para el servicio de comunicaciones unificadas en Elastix 2***, que propone implementar en el sistema de monitorización un mecanismo que permita la continuidad de la información ante algún quiebre del canal de comunicación inalámbrico.
- ***Mitigación de interrupciones en un servidor de contenido multimedia basado en tecnología IPTV***, que propone controlar la interrupción en ambientes inalámbricos entre el servidor de contenidos multimedia y el set top box, y este envíe dicha señal hacia una TV.

En cuanto a trabajo dentro del grupo de investigación en Modelos de Procesos de Software de la ESPE se ha colaborado con el proyecto:

- ***Laboratorio industrial en Ingeniería de Software Empírica***, cuyo objetivo es mejorar la comprensión del proceso de desarrollo de software en la industria, utilizando ambientes experimentales controlados y estudios empíricos que

permitan obtener conocimiento sobre el comportamiento de las tecnologías de software en diferentes entornos.

En cuanto a trabajo del grupo de investigación de televisión digital ESPETV se ha colaborado en los proyectos:

- ***Diseño y desarrollo de una aplicación de contenidos interactivos para tv digital basada en el middleware Ginga del sistema brasileño***, se centra en el estudio y desarrollo de contenidos interactivos utilizando el middleware *Ginga* y su motor de presentación *Ginga-Nested Context Language* (*NCL*).
- ***Video streaming para la visualización en tiempo real de la funcionalidad de una aplicación interactiva transmitida en Broadcast Transport Stream***, se centra en la generación de un enlace entre el usuario y el laboratorio de televisión digital mediante su plataforma de usabilidad permitiendo realizar pruebas de funcionalidad de las aplicaciones interactivas en tiempo real.

En relación a la definición y propuesta de proyectos de investigación a ejecutarse se ha colaborado en:

- ***Monitoreo de un invernadero mediante el uso de Wireless Sensor Network (WSN) y tratamiento de flujos de datos mediante técnicas adaptativas a cambios del entorno***, que busca desarrollar un sistema que permita la captura de la información de humedad relativa, temperatura y suelo por medio de sensores en un invernadero, para almacenarlos y procesarlos mediante minería de datos con la finalidad de establecer patrones que apoyen en la toma de decisiones.

1.5. Estructura de la memoria

El trabajo que hemos llevado a cabo se presenta en seis capítulos claramente definidos en esta memoria. Cada uno corresponde a un aspecto del estudio de la investigación.

En el capítulo 2, se presentan: los avances del video streaming de video, arquitectura, servidores y protocolos, sobre redes inalámbricas; la necesidad de plantear un mecanismo que permita mitigar los impactos de las interrupciones del servicio de video streaming; las herramientas disponibles a utilizarlas en el desarrollo del mecanismo; y el modelo basado en patrones software del servicio video streaming.

En el capítulo 3 se presenta: el modelo basado en patrones software de diseño para Android utilizando agentes que se implantan en una aplicación nativa.

En el capítulo 4 se presenta la iteración del modelo anterior para incluir: experiencia inmersiva y sensado móvil; y la publicación de archivos *Keyhole Markup Language* (KML), para representar la información, tratada por el agente inteligente, con herramientas para visualizar múltiple cartografía como: Google Earth o Google Maps.

En el capítulo 5, sobre una plataforma Web multiplataforma para video en tiempo real, se establece el modelo base para controlar interrupciones de una sesión de Web videoconferencia y video streaming de video, debido a una disrupción del canal de comunicación inalámbrico. Se mantuvo el modelo base en forma iterativa con incrementales en sus dos subsistemas: establecimiento y externo. Cada uno de estos subsistemas se desarrolló con base en patrones software de diseño y funcionalidades adicionales para cumplir con el objetivo de proveer control de interrupción y continuidad del servicio de Web videoconferencia y video streaming de video.

En el capítulo 6 se presentan las conclusiones de este trabajo y se comentan algunas líneas de trabajo que consideramos serán interesantes abordar.

Por último, se presenta el glosario de términos empleados a lo largo de la memoria, las referencias bibliográficas consultadas.

2. Problemática de video streaming y videoconferencia inalámbricos

En este capítulo se presenta en primer lugar el desarrollo espectacular de las redes inalámbricas, en especial WiFi, y los avances tecnológicos de los dispositivos inalámbricos en los últimos años por el consumo de nuevos productos, como servicio de video streaming. Se estudia el auge sensacional de los servicios de video streaming y videoconferencia móviles, y se introduce en las arquitecturas que manejan para brindar QoE. Se especifican los inconvenientes que producen las interrupciones de servicio de video streaming y videoconferencia y los retos planteados para bosquejar un mecanismo que permita mitigar el impacto de una interrupción. Finalmente se diseña un modelo basado en patrones software para el servicio de video streaming.

2.1. Tecnología de redes y dispositivos

Las redes inalámbricas están emergiendo para proveer a los usuarios mejores servicios basados en las ventajas que presentan como: movilidad, flexibilidad, escalabilidad, velocidad, simplicidad y costos reducidos de implementación. Esto se debe al gran avance tecnológico que ha tenido las infraestructuras inalámbricas como redes de área personal inalámbrica (*Bluetooth*), *Home Radio Frequency* (HomeRF), *Zigbee*, *WiFi*, LTE. En la Figura 2.1 se visualiza la clasificación de las redes inalámbricas en varias categorías, de acuerdo al área geográfica desde la que el usuario se conecta a la red, denominada área de cobertura.

La tecnología WiFi define una red de área local inalámbrica (*WLAN*, del inglés *Wireless Local Area Network*) que cubre un área equivalente a la red local de un hogar o empresa, con un alcance aproximado de cien metros. Permite que los dispositivos inalámbricos que se encuentran dentro del área de cobertura puedan conectarse entre sí. Es ampliamente utilizada a día de hoy y se ha optimizado su rendimiento y eficiencia espectral [34], que ha permitido ejecutar aplicaciones de contenido multimedia. Prácticamente todos los teléfonos móviles implantan WiFi. Proporciona movilidad; facilidad de instalación y configuración; y relativas altos anchos de banda [35], tanto en la banda de 2,4 como 5 GHz.

A pesar de la expansión actual de la versión *Institute of Electrical and Electronic Engineers* (*IEEE*) 802.11n, la industria ya trabaja en nuevos productos y dispositivos basados en el estándar IEEE 802.11ac, creciendo un 30% respecto al año pasado (Figura 2.2). Este estándar dispondrá de tasas de transmisión de al menos algunos Gbps para la comunicación multimedia entre los clientes con dispositivos inalámbricos con una alta fiabilidad y una cobertura uniforme [36] [37]. El nuevo estándar IEEE 802.11ad está pensado para comunicaciones directas de gran velocidad y corto alcance entre equipos como ordenadores, móviles, tabletas, discos de red y televisores, tanto para vídeo en video streaming en alta definición (HD, del inglés *High Definition*) o ultra alta definición (UHD, del inglés *Ultra High Definition*) sin cables como para pasar grandes cantidades de datos de forma inalámbrica entre por ejemplo un disco duro y un ordenador.

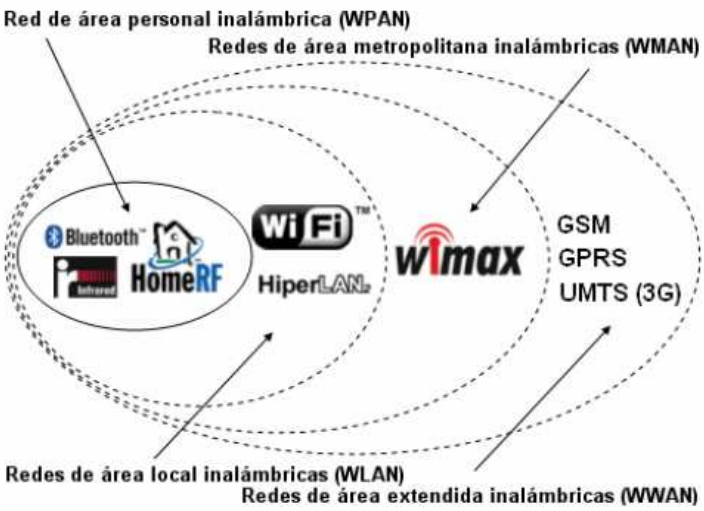


Figura 2.1: Redes inalámbricas por área de cobertura

La incorporación de nuevas tecnologías, tanto en los teléfonos móviles como en las redes inalámbricas, ha permitido el despliegue de nuevos servicios de video streaming y comunicación en tiempo real, gracias a las altas tasas de transmisión y capacidades de memoria y buferización disponibles para la implementación de estos nuevos servicios [38] [39]. Esto ha configurado un nuevo paradigma social, cultural y educativo; ya que no solo aportan con la movilidad sino también con la conectividad, ubicuidad y permanencia [40]. La insaciable demanda por dispositivos inalámbricos como: teléfonos inteligentes; tabletas y otros dispositivos, y la computación ubicua están generando una enorme cantidad de datos en las redes inalámbricas; esto se debe a la capacidad de comunicación, procesamiento y almacenamiento por parte de los dispositivos inalámbricos y su facilidad para integrarse con esta redes [41].

El *Cisco Visual Networking Index* (*VNI*) predice que el tráfico global en redes móviles se multiplicaría 18 veces en el periodo de 2011 a 2016, alcanzando 10.8 exabytes por mes [42]. En paralelo, el uso de WiFi para acceso al Internet está creciendo exponencialmente en la medida en que más dispositivos se habilitan con capacidad WiFi. El número de sitios públicos con acceso inalámbrico a Internet se expande y la aceptación del usuario se incrementa [43].

Otro paradigma que se encuentra en formación es el llamado Internet móvil, apoyado en la integración de tecnología de las redes inalámbricas, dispositivos

inalámbricos inteligentes (teléfonos o celulares inteligentes) y funcionalidades de cómputo móvil [44]. Estos logros proporcionan a los usuarios móviles acceso ubicuo a la Internet mediante un sistema convergente basado en redes de acceso heterogéneas para proporcionar servicios móviles de alta calidad [45].

El mercado de teléfonos inteligentes en todo el Mundo creció un 13,0% por año al segundo trimestre de 2015, con 341,5 millones de equipos vendidos, de acuerdo con datos de la *International Data Corporation* (*IDC*) de la estadística *Worldwide Quarterly Mobile Phone Tracker* (*WQMPT*). Android domina el Mercado con una participación de 82,8%, en el segundo trimestre de 2015, iOS presenta un descenso en un 22,3% respecto al trimestre de 2014, Windows Phone experimentó un descenso intertrimestral del 4,2% y Blackberry OS, sigue disminuyendo en el crecimiento a nivel mundial. La tabla 2.1 describe las características de las diferentes versiones del SDK de Android que han salido al mercado.

Tabla 2.1: Características de las versiones SDK de Android

Versión	Nombre	Características
1.0	Apple Pie	Lanzado en septiembre 2008. Primera versión de Android. Navegador Web con soporte de múltiples ventanas. Soporte básico de cámara de fotos. Mensajería instantánea. Reproductor de música. Soporte para teléfonos con LED. Marcación por voz. Conectividad WiFi y Bluetooth. Aplicaciones básicas. Nunca se utilizó comercialmente.
1.1	Banana Bread	Lanzado en febrero 2009. Versión que usaron para corregir errores de la primera versión. Adiciona detalles y reseñas sobre lugares y negocios en Google Maps. Nueva pantalla para manos libres. Mostrar y ocultar teclado. Guardar archivos adjuntos en los mensajes.
1.5	Cupcake	Lanzado en abril 2009. Basado en Linux kernel 2.6.27. Rediseño completo de todos los elementos de la interfaz. Transiciones animadas entre ventanas. Intérprete JavaScript. Posibilidad de personalizar los widgets mostrados en la pantalla de inicio. Añade la posibilidad de grabar y reproducir vídeos.
1.6	Donut	Lanzado en septiembre 2009 Basado en Linux kernel 2.6.29 Adiciona el Quick Search Box, una caja de búsqueda en la pantalla de inicio con autocompletado y capacidad de aprendizaje.

		Mejora la velocidad de la cámara. Posibilidad de conectarse a redes VPN, 802.1x. Nuevo motor de texto a voz.
2.0 - 2.1	Eclair	Lanzado en octubre 2009. Basado en Linux kernel 2.6.29 Rediseñó la interfaz del navegador, contando ahora con soporte para distintas características de HTML5. Mejoras en el teclado virtual. Galería 3D, al estilo Cover Flow. Nuevas aplicaciones de reloj/tiempo y noticias. Google Goggles. Mejoras en la duración de la batería.
2.2 – 2.2.3	Froyo	Lanzado en mayo 2010. Basado en Linux kernel 2.6.32. Soporte WiFi IEEE 802.11n. Soporte Flash 10.1 y Adobe AIR 2.5 Soporte de la API gráfica OpenGL 2.0 Creación de un compilador JIT que mejora entre 2 y 5 veces en Rendimiento frente a Eclair. Incorporación del mismo motor de Javascript V8 de Chrome. Opciones avanzadas de gestión energética
2.3 – 2.3.7	Ginger Bread	Lanzado en diciembre 2010. Basado en Linux kernel 2.6.35. Soporte de resoluciones de hasta 1.280x760. Soporte para: pantallas extra grandes. Reproducción de videos y decodificación de audio AAC. Teclado multitáctil. Soporte nativo para telefonía. Soporte para sensores como giroscopio y barómetro. Administración de la energía mejorada.
3.0 - 3.2.6	Honey Comb	Lanzado en febrero 2011. Basado en Linux kernel 2.6.36. Sistema multitarea mejorado. Soporte para tabletas, video chat, redes WiFi. Se añade soporte para una gran variedad de periféricos y accesorios con conexión USB.
4.0 – 4.0.4	Ice Cream Sandwich	Lanzado en noviembre 2011. Basado en Linux kernel 3.0.1. Versión que unifica el uso en cualquier dispositivo. Interfaz limpia y moderna con una nueva fuente llamada Roboto. Aceleración por hardware que permite manejar a la interfaz aumentando notablemente su rapidez. Respuesta a la experiencia de usuario. Multitarea mejorada. Gestor del tráfico de datos de internet. Mejoras en el corrector de texto. Posibilidad de realizar fotografías panorámicas de forma automática. Reconocimiento de voz del usuario. Reconocimiento facial. Lápiz táctil.
4.1 – 4.3.1	Jelly Bean	Lanzado en julio 2012. Basado en Linux kernel 3.0.31. Optimización de las transiciones en la interfaz. Pantalla de inicio con widgets e iconos. Vista previa de las capturas fotográficas. Predicción de palabras del teclado.

		Soporte offline para introducción de texto por voz. Nuevos idiomas y mejor accesibilidad con Braille. Barra de notificaciones para devolver llamada. Lectura del inicio de un mail. Redes sociales. Mejora en el reconocimiento de voz.
4.4 – 4.4.4	KitKat	Lanzado en octubre 2013. Basado en Linux kernel 3.4. Introducción de nueva funcionalidad de seguridad Mejoras en la firma digital y cifrado de contraseñas Soporta la transmisión de datos por infrarrojos. Impresión por WiFi. Optimiza el consumo de la señal digital.
5.0 – 5.1.1	Lollipop	Lanzado en noviembre 2014. Basado en Linux kernel 3.4.108. Desbloqueo inteligente Múltiples usuarios en un dispositivo. Mayor duración de la batería. Mejor multitarea. Configuración rápida del dispositivo. Encriptación automática active para proteger la información. Menor latencia en la entrada del sonido. Soporte de accesorios de sonido a través de USB Protocolo más eficiente en el consumo de energía.
6.0	Marshmallow	Lanzado en octubre 2015. Basado en Linux kernel 4.3. Mejora en los sensores de huella digital Modo de ahorro de energía. Moderniza el modelo de permisos para los usuarios para instalar y actualizar aplicaciones. Mejoras en Google play. Respaldo con Google Drive. Mejoras en la administración de memoria.

A septiembre de 2015, la Figura 2.2 muestra que la versión del SDK de Android Lollipop sube del 18% al 21%; KitKat se mantiene inamovible ya que pasa del 39,3% al 39,2%, un cambio casi nulo; Jelly Bean del 33,7% al 31,8%, mientras Ice Cream Sandwich baja poco a poco del 4,1% al 3,7%; Gingerbread del 4,6% al 4,1% y Froyo del 0,3% al 0,2%, de acuerdo a los informes mensuales de septiembre y agosto presentados por Google.

En conclusión Android 4.4 KitKat es la versión más utilizada por los usuarios de teléfonos inteligentes, de las nuevas versiones vemos como Android 5.1 Lollipop sube del 2.6% al 5.1%, que representa una gran subida, demostrando prácticamente que toda la gama alta ya se encuentra actualizada. No se observa todavía valores de Android 6.0 Marshmallow debido a que es una versión beta, de la cual todavía no se recogen datos.

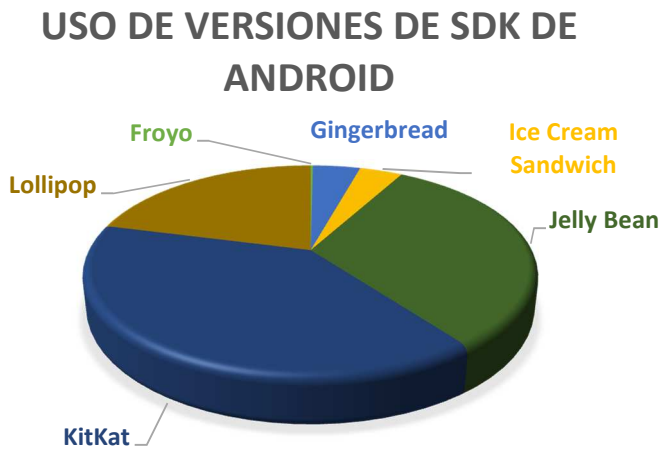


Figura 2.2: Uso de las versiones de SDK de Android en teléfonos inteligentes

2.2. Servicio de video streaming y videoconferencia

El video streaming es un servicio muy demandado actualmente [46]. Pero requieren garantizar parámetros altos de eficacia y eficiencia en las técnicas de compresión utilizando CODEC que permitan tener altas prestaciones de bajo consumo de energía, baja complejidad de codificación y decodificación y baja latencia [47] [48] [49] [50]. Muchos de los esfuerzos de investigación en relación con video streaming se han dado en tópicos de codificación y transmisión de la información multimedia, con la finalidad de lograr video streaming eficiente, robusto, escalable y de baja latencia [51]. Pero nada en relación a tratar de resolver el problema que genera una disrupción del canal inalámbrico sobre una sesión multimedia establecida.

El crecimiento exponencial de la Internet, ha dado a que el tráfico de video streaming crezca en igual sentido por la amplia disponibilidad de contenidos multimedia y la creciente capacidad de los dispositivos inalámbricos en memoria, procesamiento, batería y pantalla. Este desarrollo ha influido en la proliferación de aplicaciones que utilizan la tecnología video streaming para la implantación de sistemas como televisión por protocolo de Internet (IPTV, del inglés *Internet Proctocol Television*), red de entrega de contenidos (CDN, del inglés *Content Delivery Network*), video conferencia, entre otros, donde la demanda es por grandes capacidades de ancho de banda [52][53]. Los usuarios de teléfonos móviles consumen servicios en tiempo real como noticias,

publicidad, medicina, mensajes de advertencia, información de hospedaje y alimentación, entre otros [54] [55] [56]. Pero también demandan alta calidad de imagen para lo cual se requiere de capacidades de anchos de banda muy grandes. Con el video streaming se están explorando nuevos modelos de negocio como: plataformas de contenido y su facilidad de adaptación a los diferentes dispositivos, cambios de infraestructura en los distribuidores de contenidos, personalización de servicios video streaming, coches conectados a servicios video streaming, desarrollo de dispositivos de contenido enriquecido, entre otros [57].

En la actualidad, la implantación de sistemas IPTV supone una revolución en cuanto a la forma de ver los contenidos multimedia, para las televisiones esto conlleva la superación de los límites territoriales, una mayor diversidad de programas y otras actividades interactivas [58]. A pesar de ello, esta tecnología se ha ido extendiendo de forma muy lenta debido al coste que les supone a las empresas mantener unos servidores desde los que realizar un video streaming de calidad, teniendo en cuenta el ancho de banda necesario para toda la audiencia potencial.

El consumo de contenidos multimedia a través de WiFi genera un importante volumen de tráfico en la red y la tendencia sigue en aumento. Los grandes avances tecnológicos en capacidad de procesamiento, tecnologías de compresión y dispositivos de almacenamiento con gran ancho de banda han permitido entregar servicios multimedia en tiempo real sobre redes WiFi. Esta comunicación de video streaming sobre redes WiFi ha progresado mucho en los últimos años variando desde bajar un archivo para reproducirlo a tener varias técnicas adaptables al medio y desde el uso directo de la infraestructura de red, al diseño y uso de arquitecturas superpuestas.

La tecnología video streaming permite la distribución de contenidos multimedia (audio, video) a través de una red de datos (como puede ser red WiFi) en forma continua y en tiempo real, sin necesidad de descargar el archivo [59]. El proceso de transmisión de video streaming tradicional exige dos canales o flujos muy bien diferenciados: control y envío de los datos multimedia (Figura 2.3).

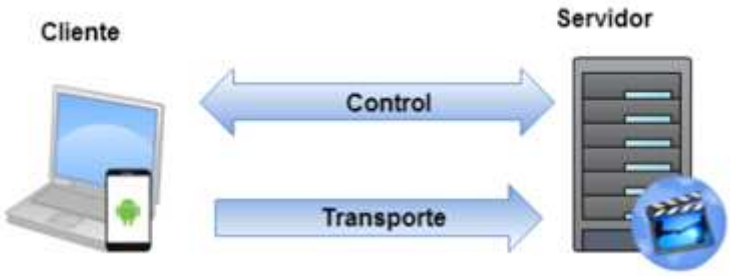


Figura 2.3: Flujo de control y transporte sesión multimedia

El primero gestiona el establecimiento de sesión, la comunicación y las interacciones que el usuario realiza con el servidor. El segundo, que es el canal de transporte, envía los paquetes de contenido multimedia [60].

Una serie de tareas e intercambio de mensajes se presenta entre el cliente y el servidor para establecer la sesión. Establecida la sesión, inicia la transmisión de los contenidos, que requieren pasar por un proceso de adaptación para poder ser enviados a través de la red WiFi. Esta adaptación es la segmentación de la información multimedia en paquetes de un tamaño adecuado para su distribución por la red WiFi, intentando adaptar a una filosofía de transmisión de conmutación de paquetes [61]. Al mismo tiempo, se añade información para mantener el control y las pérdidas que se puedan ocasionar, debido a es posible que cada paquete viaje por un camino diferente, con diferentes condiciones de transmisión. La integridad de los paquetes recibidos se analiza consultando la información de las cabeceras, consiguiendo el acoplamiento del contenido recibido para su reproducción [60], [62].

Conforme el cliente recibe los paquetes de contenido multimedia, procede a su almacenamiento en un buffer. Cuando el buffer está lleno, el cliente comienza la reproducción y al mismo tiempo continúa recibiendo nuevos paquetes de información. Una de las funciones importantes del buffer es de proveer una amortiguación a efectos de pequeñas variaciones en las condiciones de la red WiFi, de manera que si se produce una bajada en el ancho de banda disponible para la transmisión, el cliente puede continuar con la reproducción de los paquetes almacenados en el buffer [63]. Si los problemas de disponibilidad de ancho de banda persisten, dicho buffer se vacía y en

consecuencia la reproducción se detiene. Desde el punto de vista de la calidad percibida por parte del usuario, una reproducción continua genera tener condiciones de transmisión estables [61].

Los servicios video streaming se emplean hoy en día en los diferentes medios de comunicación para transmitir eventos en directo o en diferido, y no olvidarse de videoconferencia y educación.

El transporte de video en vivo o almacenado previamente es la parte más importante de estos servicios multimedia, entre ellos tenemos:

- *Descarga tradicional*, el cliente descarga toda la información a disco y después la reproduce.
- *Descarga progresiva* (*pseudo-video streaming*), el cliente reproduce la información multimedia según la va descargando a disco o a memoria (YouTube, Google Videos…).

Existen dos modalidades para implantar un servicio video streaming:

- *Servicio bajo demanda*, envía el contenido que solicite el cliente en cualquier instante. Este servicio lo provee un servidor de video streaming, quien atiende las peticiones de visualización del contenido multimedia, mediante la entrega de flujos para ser visualizados en el reproductor del cliente. El usuario tiene la posibilidad de controlar el flujo.
- *Servicio en directo*, está orientado a la multidifusión, el usuario accede a un único flujo de información en cualquier momento en el que se encuentre la emisión. Carece de interactividad.

2.2.1. Arquitectura del servicio de video streaming

El servicio de video streaming fundamenta su arquitectura básica en un modelo de aplicación distribuida cliente-servidor.

En la Figura 2.4 se observa un esquema de la arquitectura con todos los elementos que conforman el servicio de video streaming.

El *sistema de producción* se encarga de capturar los contenidos en vivo de audio y/o video y los transmite con el formato adecuado al servidor. La captura de los contenidos se hace a partir de elementos como micrófonos o cámaras. Este sistema se encuentra conformado tanto de hardware y software con el objetivo de comprimir los contenidos multimedia, para que resulten aptos para su transmisión a través de los protocolos de transporte. Para el tratamiento de los contenidos originales interactúa el CODEC que puede ser de audio y/o video.

El *sistema de almacenamiento* se encarga de almacenar los contenidos, que pueden ser solicitados por los clientes. En el video bajo demanda es uno de los componentes principales.

El *servidor* se encarga de recibir y procesar peticiones de los clientes, establecer la sesión mediante el intercambio de parámetros y procesar el envío de los contenidos multimedia desde sistema de producción o sistema de almacenamiento.

El *reproductor* se encarga de la interacción con el servidor y ejecuta las funciones de establecer la conexión, procesar los paquetes y reconstruir el flujo de datos recibidos.

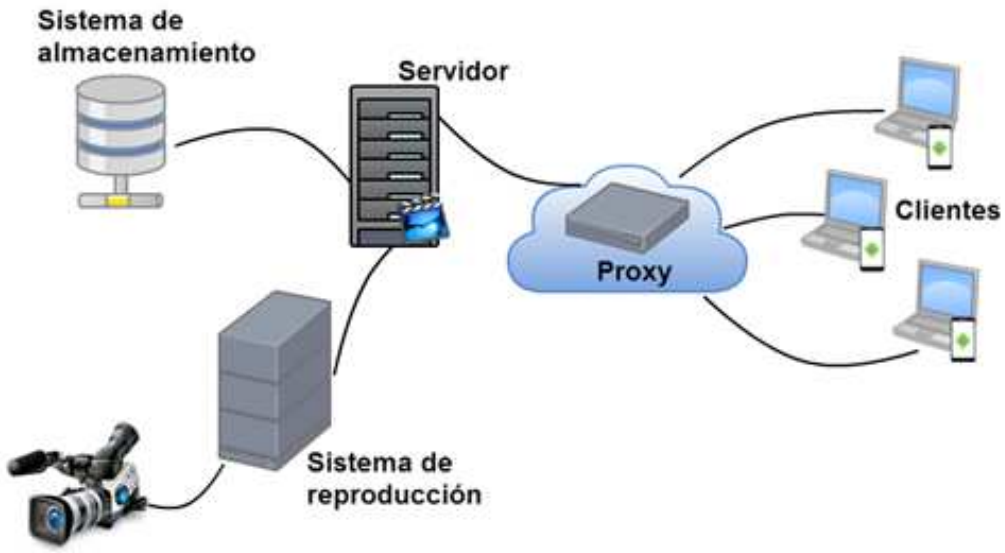


Figura 2.4: Arquitectura básica servicio video streaming

El *Proxy* se comporta como un *Proxy* multimedia y se encarga de mantener la comunicación con el cliente durante la reproducción del video, estado de la sesión e interacción con el usuario [64], [65].

Algunas arquitecturas que emplea las tecnologías video streaming se las detalla a continuación [60], [62]:

- *Arquitectura típica*, usa la arquitectura cliente-servidor y los protocolos más usados son:
  - Para el escenario sin control sobre la transmisión tenemos *Hypertext Transfer Protocol* (*HTTP*).
  - Para el escenario con control sobre la transmisión tenemos:
    - En el nivel de aplicación: RTSP responsable de la entrega de datos en tiempo real, no orientado a conexión. El control y reenvío de datos es responsabilidad de *Transmission Control Protocol* (*TCP*), *Microsoft Media Server* (*MMS*), *Real Time Messaging Protocol* (*RTMP*) y *Real Time Media Flow Protocol* (*RTMFP*).
    - En el nivel de transporte: *Real Time Transport Protocol* (*RTP*), *User Datagram Protocol* (*UDP*) y TCP.
- *Arquitectura sin servidor* (*Server-Less*), no presenta un servidor de audio-video, el archivo se le proporciona al cliente mediante un servidor Web (pseudo-video streaming o Fast-Start). Usa TCP y HTTP.
- *Arquitectura sin cliente*, no hay una aplicación cliente. Simula el funcionamiento de un servicio bajo demanda con flujo de datos en directo. Para visualizar el contenido multimedia se utiliza un applet java o algún plugin.

En todas estas arquitecturas ninguna incluye un mecanismo para reconectar automáticamente la sesión de comunicación ante una interrupción del canal inalámbrico. El servidor es el encargado de transmitir nuestro programa contenido multimedia (programa de televisión, archivo de video…) usando la tecnología video streaming. Para entender mejor cómo funciona en el contexto de la tecnología video streaming, vamos a explicarlo como un proceso dividido en tres etapas: adquisición,

codificación-transcodificación y entrega. Durante el proceso de adquisición se captura la señal en directo (cámara o tarjetas de captura de audio y video) o se la recibe de algún sitio Web que permita compartir videos, para después codificar la señal. La primera codificación se lleva a cabo en el origen siguiendo los estándares del mercado, en este punto es donde la señal tiene la máxima calidad [66]. Existen algunos software de codificación, los más comunes son; *Wirecast*, *Viewcast*, *Digital Rapids*, *Flash Media Live Encoder*… En el proceso de transcodificación se descomprime la señal codificada (la señal entra al servidor video streaming) y después se codifica de nuevo a los diferentes formatos. La señal se optimiza para que llegue sin problemas de retransmisión a los diferentes dispositivos. Finalmente en la entrega la señal llega al usuario final en cada una de las calidades y formatos disponibles a los diferentes dispositivos. Esta entrega se la realiza con cualquiera de los protocolos para servicio de video streaming.

Los servidores de video streaming deben procesar datos multimedia con ciertas restricciones temporales para prevenir fallas (llamadas *jerkiness* en video y *pops* en audio), para poder ofrecer servicios de calidad [67]. Además deben soportar comandos tipo VCR 59 que permitan parar, poner en pausa, adelantar o retroceder el video y entregar el audio y video sincronizados [68].

Un servidor típico de video streaming consta de:

- *Comunicador*: contempla la capa de aplicación y los protocolos de transporte implementados en el servidor.
- *Sistema Operativo*: debe soportar aplicaciones en tiempo real a más de los servicios típico.
- *Sistema de almacenamiento*: debe soportar almacenamiento y retiro continuo de medios.

Uno de los más utilizados como servidor de video streaming es VLC media player, del proyecto VideoLAN. VLC media player es un framework y reproductor multimedia libre y de código abierto, multiplataforma. Soporta la mayoría de archivos multimedia, así como DVD, Audio CD, VCD y diversos protocolos de transmisión. Utiliza la biblioteca CODEC *libavCODEC* del proyecto *FFmpeg* para manejar los muchos formatos soportados

por esta, y emplea la biblioteca de descifrado DVD *libdvdcss* para poder reproducir DVD cifrado.

El desarrollo de los protocolos de comunicación y transporte, que mejoran y agilizan el proceso de transmisión de datos, han permitido que se implementen servicios de video streaming. Los protocolos de transporte se encargan del control de errores, la secuenciación y el control de flujo en el servicio de video streaming.

Se presenta dos protocolos TCP [69] que ocasiona un retraso muy elevado, debido a procesos de retransmisión que los servicios en tiempo real no pueden asumir; y UDP [70] que permite el envío de datagramas sin control de flujo ni necesidad de confirmación de entrega, evitando la demora del proceso con interrupciones por errores en el protocolo.

Los protocolos de comunicación permiten gestionar el establecimiento y sesión mediante RTSP y el transporte y control mediante RTP y *Real Time Control Protocol* (*RTCP*).

RTSP [8], es un protocolo que se encarga de mantener y controlar la sesión entre el cliente y el servidor de streaming. Actúa únicamente en la parte de control del envío de datos en tiempo real, ya que los datos viajan por canales diferentes. Permite establecer una comunicación entre el cliente y el servidor para ejecutar interacciones (pausa, avance, retroceso) durante la reproducción e intercambio de mensajes por medio de un conjunto de métodos.

RTSP es independiente de la capa transporte y distingue tres estados: inicial, listo (estado posterior a la inicialización de la sesión) y emitiendo (estado en el que se lleva a cabo la distribución de los contenidos). En la Figura 2.5, se observa la transición entre un estado u otro mediante la ejecución de los métodos.

Un ejemplo del proceso de establecimiento de sesión se observa en la Figura 2.6, además del intercambio de mensajes propios del protocolo RTSP, también observamos el envío de la información multimedia mediante el protocolo RTP y el envío de información de control por parte del cliente mediante protocolo RTCP.

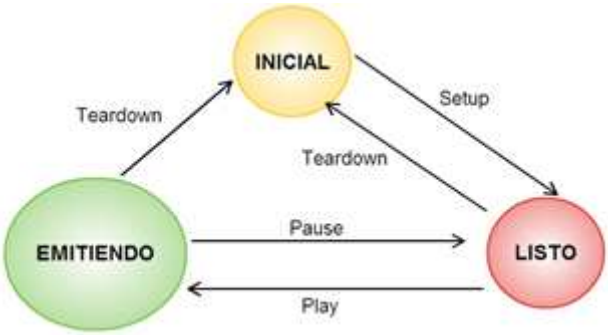


Figura 2.5: Diagrama de estados para RTSP

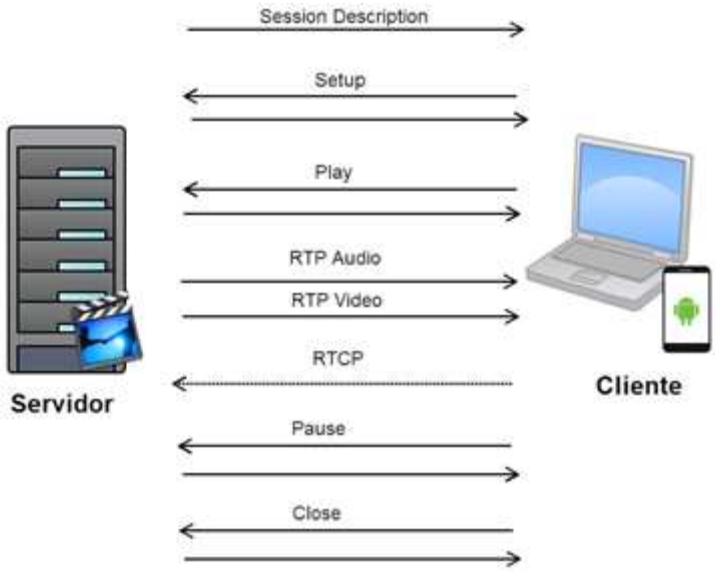


Figura 2.6: Establecimiento de conexión en RTSP

RTP [71], es un protocolo para el transporte extremo a extremo de datos (audio y video) en tiempo real sobre la Red. No garantiza que los paquetes lleguen a su destino, producto de la variación de las características de la Red en el transcurso de la comunicación. Generalmente trabaja sobre UDP, por lo que necesita apoyo de protocolos asociados, como RTCP, para informar y realizar cierto control sobre el estado del canal de comunicación. No define un mecanismo para asegurar la calidad de servicio, porque solamente se centra en el transporte de los datos.

RTCP [72], funciona en conjunto con RTP y su objetivo es la monitorización del envío de paquetes RTP, mediante la transmisión periódica de paquetes de control. Su

función principal es de entregar información estadística de la conexión acerca de calidad de servicio (pérdida de paquetes, retraso, jitter), que puede ser empleada por aplicaciones para ajustar la codificación y otros parámetros.

Session Description Protocol (*SDP*) [73], es un protocolo que se emplea para describir los parámetros de inicialización de sesiones multimedia. Define parámetros que permite cubrir aspectos como anuncio de sesión, invitación a sesión y negociación. No se encarga de entregar los contenidos (datos), sino de entablar una negociación entre las entidades que intervienen en la sesión como tipo de contenido, formato y demás parámetros asociados.

2.2.2. El servicio de videoconferencia

La videoconferencia es un servicio de telecomunicación multimedia, que permite la comunicación e interacción en tiempo real entre dos personas o grupo de personas que se encuentran geográficamente distantes [74]. Dependiendo de la tecnología que se utilice, la videoconferencia logra ser completamente interactiva y crea un ambiente como que todas las personas participantes de la videoconferencia se encontrasen en la misma ubicación física [75]. La forma genérica de clasificar los sistemas de videoconferencia considera el tipo de enlace [76]. Dentro de esta clasificación se encuentran:

- *Sistema punto a punto*: la comunicación se realiza entre dos puntos remotos.
- *Sistema multipunto*: la comunicación se realiza entre dos puntos o más sedes enlazadas.

Los componentes básicos que conforman un sistema de videoconferencia son los siguientes:

- *Red de comunicaciones*: es donde se consolida el sistema de videoconferencia, proporciona una comunicación digital bidireccional. La selección de la red de comunicación para prestar el servicio de videoconferencia depende de los requisitos del usuario.

- *Sala de videoconferencia*: es el espacio acondicionado para alojar a los participantes de la videoconferencia. Considera cuatro componentes: ambiente físico, sistema de video, sistema de audio y sistema de control.
- *CODEC*: es el que provee de los formatos de audio y video necesarios para establecer la videoconferencia.

Hasta hace unos años, la videoconferencia tendía a encajar en uno de estos dos ámbitos: empresas con grandes presupuestos que tenían grandes salas de conferencias especializadas, que recuerdan a pequeños estudios de difusión; y organizaciones más pequeñas que tenían que buscar soluciones más rentables que afectaban a la calidad de la imagen y el audio. En ambos casos, las deficiencias en la facilidad de uso y la fiabilidad conllevaban frecuentes llamadas telefónicas al equipo de apoyo en busca de asistencia.

Uno de los claros culpables durante esta primera generación de soluciones de videoconferencia era la conexión a Internet, ya que aquellos que tuvieran mala conexión tenían que descifrar los diálogos irregulares con la visión borrosa de los que hablaban [77].

Con estas restricciones en mente, se pretendía conseguir llamadas que fueran lo suficientemente buenas; después de todo, el propósito era ahorrar costes al eliminar la necesidad de viajar por reuniones presenciales.

En la actualidad, la videoconferencia de alta calidad no está al alcance de unos pocos, sino de unos muchos. La mejora del acceso a Internet de banda ancha de alta velocidad, las redes WiFi y el 4G han favorecido la creación de servicios de consumo como *Skype*, *Google Hangouts* y *WebRTC*, mientras que las restricciones presupuestarias han intensificado aún más la necesidad de un modelo de negocio virtual para organizaciones de todo tipo de tamaños y formas.

El futuro de la videoconferencia puede estar dentro del propio navegador, WebRTC es un estándar abierto que forma parte de la especificación de HTML5 y que permite la comunicación de video y audio de alta calidad a través de la Web sin necesidad de utilizar plugins adicionales para poder realizar esta tarea.

2.2.3. Degradación de la calidad de experiencia de usuario

Es importante en todo servicio de video streaming sobre redes inalámbricas no solo tomar en cuenta la valoración del rendimiento de la Red sino también tomar en cuenta la percepción, perspectivas y hábitos del usuario [78]. El problema de entregar QoE en redes inalámbricas ha sido abordado desde diferentes aristas como disponer de eficientes técnicas de compresión del audio y video, tecnologías alternativas como video streaming adaptativo, entre otros [79]. La QoS de la comunicación inalámbrica no es el único parámetro considerado para determinar si un servicio de video streaming establecido es óptimo. Es necesario tener un punto de vista del usuario, quiere decir, valorar la percepción (*QoE*) que el usuario tiene sobre el servicio. Estos aspectos pueden ser el tamaño de la pantalla y resolución del dispositivo móvil, CODEC utilizado y capacidad del búfer de datos ante las tasas de velocidad de carga durante la reproducción del video [80]. La QoS permite administrar los efectos que tiene el fenómeno de la congestión sobre el rendimiento del servicio de video streaming, para esto se hace uso de servicios integrados y servicios diferenciados que trabajan sobre los diferentes flujos de datos o sobre usuarios, como factor de métrica a nivel de capa de red, la QoS mide la cantidad de paquetes perdidos, el retraso y la variación del mismo (Jitter) [81].

QoE evalúa la calidad del vídeo desde dos perspectivas:

- Desde el punto de vista del usuario: es decir, la percepción de la calidad del vídeo recibido en el cliente a la hora de su reproducción y visionado. Para este caso la degradación de la calidad se puede presentar de muchas maneras: por el efecto bloque, la pixelación, congelado de la imagen, entre otros. La medida de la calidad de los contenidos del vídeo recibido en el cliente puede realizarse mediante métricas objetivas o mediante métricas subjetivas. Las métricas objetivas están basadas en el uso de algortimos y las más empleadas son: PSNR y SSIM. Ambas requieren acceso a los contenidos originales para poder cuantificar la calidad de la imagen recibida. Por su parte, las métricas subjetivas se basan en las

opiniones de los usuarios frente a los estímulos de vídeo recibidos. El estudio de la calidad de vídeo mediante evaluaciones subjetivas supone un consumo de tiempo elevado, por lo que, habitualmente, se recurre a métricas objetivas. A pesar del dominio de las métricas objetivas PSNR y SSIM, existen otras, como MOVIE [82], que obtienen un elevado grado de correlación con las métricas subjetivas. Su principal inconveniente, a día de hoy, es el elevadísimo coste computacional durante su cálculo.

- Desde el punto de vista de la red: además de las métricas anteriores de calidad percibida, e independientemente del tipo de CODEC empleado, tenemos que tener en cuenta la calidad de la transmisión reflejada a través del nivel de servicio de la red (es decir, si la conexión es capaz de transportar el vídeo). Los parámetros de ancho de banda, retraso, jitter y pérdidas serán determinantes para decidir si la red está preparada para ofrecer tráfico en tiempo real.

Hoy en día aún existen una serie de limitaciones tecnológicas para ofertar un servicio de audio/vídeo de alta calidad a través de redes inalámbricas. La dinámica de las redes tipo best-effort en términos de variaciones de ancho de banda y retrasos hace que sea un problema proporcionar una buena calidad en las transmisiones video streaming. Algunas técnicas de codificación implementan métodos de corrección de errores [83].

Otra opción que podemos plantear es mejorar la arquitectura de red: podemos sobredimensionar el ancho de banda del enlace y esperar que el retraso, el jitter y las pérdidas no sean demasiado elevados (sin garantías). También podríamos solicitar la retransmisión de aquellos paquetes perdidos, pero aumentaríamos el retraso, empeorando aún más la calidad. Necesitamos, por tanto, otras soluciones para garantizar cierta QoS en los servicios de distribución de vídeo a través de redes inalámbricas. Por ejemplo, algunos programas utilizan técnicas a nivel de aplicación, como son el empleo de búffers o algoritmos de codificación resistentes a pérdidas, buscando mitigar los efectos del retraso, el jitter y las pérdidas.

Una de las causas más frecuentes de los problemas que ocurren para el servicio de video streaming es una conexión a la red WiFi débil o intermitente. Las repeticiones

de almacenamiento en buffer o las cargas frecuentes, los problemas para iniciar la sesión del servicio de video streaming, los mensajes de error que indican que no es posible conectarse a la red WiFi o problemas para reproducir la sesión en un dispositivo, generalmente indican que la conexión a red WiFi es lenta o sufre interrupciones.

Estos problemas afectan fuertemente a los diferentes tipos de servicio de video streaming que en la actualidad se utilizan:

- En directo (live), similar a un canal de televisión.
- Bajo demanda (on-demand), similar a un reproductor de vídeo.
- Casi bajo demanda, simula el funcionamiento de un servicio bajo demanda con flujos de vídeo en directo.
- Todos ellos utilizan técnicas de compresión de vídeo como MPEG-2/4, H.264/AVC, VP8 son una parte crítica del servicio de video streaming sobre redes WiFi [84], [85], [86], [87].

En las redes WiFi, se debe cumplir con características y requisitos de servicio para soportar servicio de video streaming como alto nivel de calidad, velocidad, tasa de bits, tasa de error máximo tolerable de paquetes y límites de retraso [88].

Factores que provocan las interrupciones en redes WiFi

La transmisión de datos en redes WiFi presenta muchos factores que provocan pérdidas frecuentes de paquetes y quiebres de cobertura de radio, generando interrupciones en los usuarios de dispositivos inalámbricos.

Los servicios de video streaming multimedia en la actualidad requieren de altas velocidades de transmisión de datos y grandes anchos de banda, esto es un serio problema en la comunicación inalámbrica por su baja fiabilidad y fluctuaciones del ancho de banda que conducen a la degradación de la calidad del contenido multimedia de manera significativa [89]. Este escenario es uno de los primeros problemas que se genera al transmitir paquetes de video atrasados y por lo tanto tenemos degradación tanto en el rendimiento de la red como la calidad del video.

Otros factores que modifican las condiciones del canal de comunicación inalámbrico en el tiempo son la interferencia, fading y movilidad [90], donde es importante garantizar el aprovisionamiento de la calidad de servicio de extremo a extremo, muy necesario para la reproducción continua del contenido multimedia en tiempo real.

Con respecto al ancho de banda disponible por el canal de comunicación inalámbrica, tenemos dos casos a ser analizados:

- Si el ancho de banda disponible del canal de comunicación inalámbrica es menor a la tasa de transmisión del emisor, se presenta un escenario de congestión con pérdida de paquetes y una caída drástica de la calidad de video.
- Si el ancho de banda disponible del canal de comunicación inalámbrica es mayor a la tasa de transmisión del emisor, se presenta un escenario de no pérdida de paquetes y una calidad de video óptima.

La transmisión de contenido multimedia mediante tecnología video streaming a través de redes inalámbricas se expone a factores que inciden en disrupciones en la sesión multimedia como:

- Variación en el tiempo de las condiciones de ancho de banda, retraso, jitter o alta tasa de pérdida de paquetes, generando pérdidas frecuentes de paquetes.
- Cobertura de radio, no siempre constante, produce interrupciones totalmente impredecibles en los usuarios móviles.
- Interferencia con otros dispositivos inalámbricos.
- Variación de la velocidad de transmisión de los datos, depende de numerosos factores como: efectos de atenuación, degradación de la señal, aparición de interferencias, entre otros.

En conclusión la transmisión de contenido multimedia mediante tecnología video streaming en redes inalámbricas enfrenta desafíos de las condiciones del canal, recursos de red limitados y la inestabilidad de la red inalámbrica conduce a problemas tales como

ancho de banda variable en el tiempo y limitado, congestión de tráfico durante la transmisión de una ráfaga de flujos.

Limitaciones en la red

En las que implantan *Internet Protocol* (*IP*) [91] sobre ambientes inalámbricos, los paquetes se envían sin garantía de servicio y los recursos de red se comparten equitativamente. Es por eso que los problemas en cuanto a la calidad de servicio video streaming en una red de datos se derivan principalmente de dos factores:

- La red de datos está basada en conmutación de paquetes. Por lo tanto, la información no viaja siempre por el mismo camino, lo que provoca jitter o pérdida de paquetes.
- Las aplicaciones multimedia tienen requisitos de tiempo real. El retraso o latencia y la pérdida de paquetes puede llegar a ser muy perjudicial.

Evidentemente, el comportamiento de la aplicación multimedia durante el proceso de transmisión en la red depende en gran medida del protocolo de transporte de red empleado.

Existen dos opciones, cada una de ellas presenta una serie de ventajas e inconvenientes frente a la otra: por un lado, el protocolo orientado a conexión TCP, y por otro lado, el protocolo no orientado a conexión UDP.

Limitaciones en dispositivos inalámbricos

A pesar del gran desarrollo tecnológico de los dispositivos inalámbricos, la limitación de sus recursos, en especial capacidad de memoria, pantalla y batería, siguen siendo un problema en comparación con equipos informáticos de escritorio, generando la necesidad de aplicaciones que permitan el ahorro de recursos y así proveer de calidad de experiencia en el usuario.

La tecnología video streaming contribuye en la transmisión de contenido multimedia eficientemente en dispositivos inalámbricos, pero no mitiga los problemas anteriormente descritos en el canal de comunicación inalámbrico.

La capacidad de movilidad del cliente, provoca que fácilmente pueda alcanzar zonas de no cobertura del canal de comunicación inalámbrico, generando disrupciones en la sesión de servicio video streaming multimedia.

El cliente desconoce si se encuentra dentro de la zona de cobertura, mientras utiliza un servicio de video streaming. Por lo que es necesario proveer de información (nivel de intensidad de señal) que le permita al cliente mantenerse dentro de la zona de cobertura, caso contrario tendrá problemas de interrupción de la comunicación inalámbrica.

En conclusión, pese a que las características técnicas han mejorado notablemente en los campos de visualización y comunicación, proveyendo de buenas experiencias en servicios video streaming, es importante cuidar este factor que afecta a la calidad de experiencia del cliente.

Factores que impactan la calidad del transporte de video streaming

Dentro de los factores que impactan la calidad del transporte de video streaming, veremos la repercusión que causan las limitaciones tecnológicas y los factores derivados del funcionamiento de la red de conmutación de paquetes, como son el retraso, el jitter o la pérdida de paquetes y su impacto en la calidad de la sesión video streaming [92] [93] [94].

La pérdida de paquetes es un hecho común en todas las redes de conmutación de paquetes y representa el porcentaje de unidades de datos que no llegan a su destino en un intervalo de tiempo específico [93]. En este tipo de redes no hay ninguna reserva de recursos en un proceso previo al envío de la información, como consecuencia, los paquetes se pueden perder debido al ancho de banda baja de la conexión inalámbrica, la atenuación de la señal [95].

Puede ocurrir que, durante periodos de congestión, algún nodo de la red no sea capaz de procesar un paquete y lo descarte, dando lugar a una pérdida. A esto añadimos que el servicio de video streaming tradicional a través de Internet emplea UDP el cual no ofrece ningún tipo de garantía en la entrega de los paquetes.

Esta pérdida de paquetes se puede recuperar aplicando diversas técnicas mediante la retransmisión de paquetes en la capa de transporte, la corrección de errores en la capa física, o el uso de los códigos en la capa de aplicación [96].

El retraso se encuentra determinado por el tiempo que transcurre desde que se generan las muestras de audio/vídeo en el extremo emisor hasta que llegan al receptor [84], [93]. Retraso extremo a extremo excesivo provoca una pérdida de paquetes, produciendo un efecto apreciable en la percepción de la calidad en videoconferencia [96].

Podemos indicar que el retraso se lo puede calcular:

$$Retraso = T_{n-1} - T_n \ (ms)$$

Donde:

T_{n-1} = tiempo de recepción del anterior paquete.

T_n = tiempo de llegada del paquete.

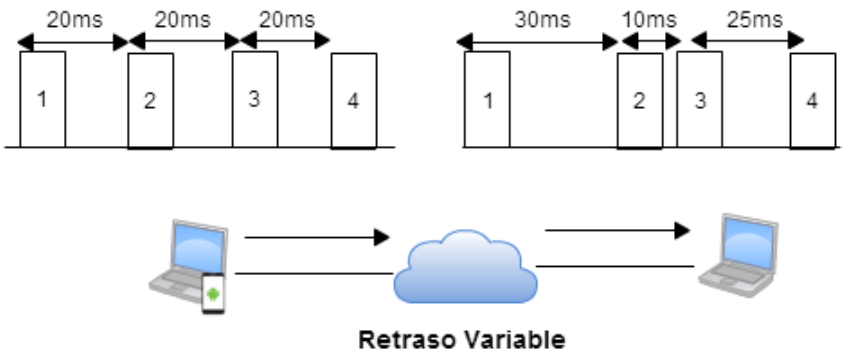


Figura 2.7: Ejemplo de afectación del jitter

El jitter, que es variación de retraso [93], es la fluctuación del retraso entre paquetes dentro del flujo de la misma conexión [84]. Únicamente está presente en las redes de conmutación de datos, ya que cada paquete puede recorrer caminos distintos entre origen y destino y por tanto puede sufrir retrasos distintos. Su efecto puede ser más perjudicial que el propio retraso. Si el paquete se retrasa más de lo debido, se dará por perdido, con la consiguiente disminución de la calidad [96].

En algunas aplicaciones el jitter se puede reducir mediante el almacenamiento temporal de los paquetes en el receptor a través de un buffer (Figura 2.7). Ese almacenamiento dura el tiempo suficiente para que los paquetes se puedan reordenar y reproducir en el orden correcto.

El buffer de supresión del jitter presenta el inconveniente del aumento del retraso en el extremo receptor. Además, si la llegada de paquetes se produce en instantes en los que el buffer está lleno, los paquetes se descartan, ocasionando una nueva fuente de pérdida de paquetes.

El ancho de banda, que es la cantidad de datos que pueden fluir a través de una conexión de red en un periodo de tiempo dado [93], es uno de los factores más importantes en la ingeniería de redes. El video streaming de audio y vídeo requiere grandes cantidades de ancho de banda y constituye uno de los puntos claves a la hora de mejorar la calidad percibida por el usuario.

En esta línea, las aplicaciones que usan tecnología de video streaming tienen como objetivo adaptar la calidad de los contenidos transmitidos al ancho de banda disponible con el fin de maximizar la experiencia de usuario.

Al presentarse una interrupción generada por una desconexión del canal inalámbrico de aproximadamente un minuto o más, la sesión de video streaming se pierde y se debe reiniciar provocando la petición de establecimiento de una nueva sesión y posterior visualización del contenido multimedia desde el inicio. Esto genera un gran malestar en el usuario del dispositivo móvil que termina por abandonar la sesión de video streaming.

Factores que impactan la QoE de video streaming

En redes inalámbricas resulta muy complicado garantizar la calidad de servicio con requisitos de tiempo real, por no tener control sobre el enrutamiento de los datos y la disrupción en una comunicación inalámbrica. Esto genera situaciones de elevada latencia y pérdida de paquetes durante la transmisión [11]. La baja fiabilidad de estas redes, produce fluctuaciones de ancho de banda que conducen a la degradación de la calidad de video de manera significativa [12].

No debemos olvidar que Internet no fue diseñada con el propósito de transportar información multimedia. Aunque podamos transmitir contenidos de audio/vídeo y datos por la misma red, no va a ser posible tratarlos de la misma manera, ya que los primeros son mucho más sensibles al retraso, al jitter y a la pérdida de paquetes por tratarse de fuentes de tráfico de tiempo real y los segundos presentan exigencias más flexibles. Es por eso que se hace necesario el uso de mecanismos adicionales para complementar el servicio best-effort que ofrece IP con el fin de obtener una buena calidad de la sesión video streaming [97]. En un sistema de comunicación, donde se encuentra implementada una sesión de video streaming, el envío de un número excesivo de paquetes sin tener en cuenta el ancho de banda disponible, genera una congestión inevitable [14]. El aprovisionamiento de la QoS extremo a extremo es necesario para el mantenimiento continuo de reproducción de video en aplicaciones multimedia en tiempo real (videoconferencia). Por lo que es muy importante evitar que las condiciones del canal inalámbrico cambien en el tiempo debido a interferencia, fading y movilidad, convirtiendo al servicio de video streaming es una tarea difícil de sostener [13], [14]. La evolución tecnológica exponencial de los dispositivos inalámbricos permite a los usuarios el consumo de nuevos servicios como video streaming. Sin embargo, la capacidad de memoria, batería y procesamiento son bajos en los teléfonos inteligentes en relación a las computadoras portátiles. Esto marca la necesidad de desarrollar aplicaciones que permitan el ahorro de las capacidades anteriormente indicadas y lograr la calidad de experiencia en el usuario [98–100]. Una interrupción en la sesión de video streaming establecida provoca que la visualización de los fragmentos, se limite solo a lo que se encuentre cargado en el buffer cliente. Si los tiempos de interrupción son muy altos, ocurre la pérdida de sesión. El usuario al darse

cuenta de la interrupción, probablemente solicite el servicio, generando nuevos tiempos de establecimiento de sesión e incrementando al tiempo total. En WiFi factores que provocan interrupciones son: banda que utiliza múltiples dispositivos de comunicación; interferencias generadas por dispositivos inalámbricos próximos; factores atmosféricos pueden interferir en la señal; limitación de ancho de banda; tráfico de red; pérdida de cobertura; potencia de emisión; protocolo de red inestable [90], [101].

2.3. Agentes de software inteligentes

Un agente inteligente, es una entidad capaz de percibir su entorno, procesar tales percepciones y responder o actuar en su entorno de manera racional, es decir, de manera correcta y tendiendo a maximizar un resultado esperado. Todo agente tiene una función u objetivo. Según [102] agente inteligente representa una entidad de software que, basándose en su propio conocimiento, realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de estos se lo requiere. Para determinar las tareas proactivas a realizar de acuerdo al dinamismo del medio ambiente en el que actúan, se comunican intercambiando mensajes y se caracterizan por su proactividad, inteligencia y movilidad. De acuerdo al punto de vista de la inteligencia artificial y según [103], agente es la entidad de software que está situado en algún medio ambiente y es capaz de realizar una acción autónoma de acuerdo al entorno con el fin de cumplir sus objetivos de diseño. Los agentes tienen que establecer por si mismos su propia manera de aplicar recomendaciones y tomar una decisión de confianza, es decir proporcionar mayor eficiencia y seguridad, para ello se basan en las siguientes propiedades: autonomía, sociabilidad, capacidad de reacción, iniciativa, benevolencia y racionalidad [103].

Un agente tiene capacidad de proceso puesto que puede descomponer una consulta en subconsultas y asociar a los distintos términos resultantes otros términos relacionados o afines. Su conocimiento del entorno le viene dado por su propio conocimiento y por el de otros agentes que se comunican con él. En todo momento debería saber a qué información acceder o a qué otro agente dirigirse para obtenerla.

Un agente puede tener también acceso a un dominio y/o información de un modelo, si se asocia con la estructura de éste.

Muchos de los sistemas existentes no tienen capacidad de colaboración, se utiliza agentes inteligentes para obtener la capacidad de cooperación entre ellos y así poder realizar y ejecutar las actividades encomendadas.

Otro aspecto importante de resaltar es *Agent Management System* (*AMS*), sistema compuesto de múltiples agentes inteligentes interactuantes, cuyo objetivo principal es resolver problemas difíciles de hacerlo por un agente individual, gracias a la interacción de uno con otro mediante mensajes en ese entorno basados en características como: habilidades sociales, proactivas y reactivas [104]. Se debe considerar que para desplegar un agente debemos tomar en cuenta todos los aspectos referidos a su comportamiento y apoyarnos en metodologías y herramientas que permitan construir aplicaciones distribuidas basadas en componentes. Es necesario entonces considerar los siguientes aspectos definidos en [105]:

- Modelar los agentes en un sistema y las interfaces de estos agentes.
- Describir la información consumida y generada por cada agente.
- Esbozar las posibles interrelaciones entre agentes.
- Especificar el contenido de la información intercambiada entre agentes, estas especificaciones deber ser interpretables por la computadora y proveer suficiente detalle para conducir al despliegue de los agentes.

Uno de los últimos desarrollos en tecnología de agentes, son los agentes móviles, se basan en el principio organizador de redes de comunicación entre ordenadores, conocido como Control de Procedimientos Remotos (RPC, del inglés *Remoto Procedure Control*) [106].

Cuando un ordenador cliente de una red (no importa su tamaño) dirige una petición al servidor de archivos para ejecutar una aplicación, el cliente debe realizar al menos dos comunicaciones: una solicitando la ejecución de un programa determinado, y otra informando al servidor que la operación se ha completado con éxito.

La alternativa a este procedimiento es la Programación Remota (RP, del inglés *Remote Programming*), consistente en acordar por adelantado qué tareas pueden realizar los clientes sin ningún tipo de verificación ni confirmación por parte de los servidores [106]. De esta forma un cliente enviaría una instrucción al servidor de archivos, y una vez allí ejecuta un programa en concreto. Este procedimiento (remoto) que es una orden realizada por el cliente pero ejecutada en el servidor (local) recibe el nombre de operación o instrucción móvil, haciendo hincapié en que se trata de una orden remota que se ejecuta localmente.

Un agente móvil puede suspender el proceso que esté realizando, transportarse a sí mismo por medio de la red y reanudar la ejecución del proceso que estaba llevando a cabo donde estime oportuno. Esta capacidad le permite al agente seleccionar la información recuperada antes de enviarla por la red, lo que evita la transferencia de grandes cantidades de información que podría ser inútil.

Existe una gran cantidad de software denominado como software de agentes para Linux, Windows y otros sistemas operativos, sin embargo la herramienta para desarrollar agentes más extendida y utilizada es JADE [107] gracias a sus buenas herramientas gráficas, documentación, soporte, *Licencia Pública General Reducida* (*LGPL*) de GNU.

JADE

JADE es un middleware que proporciona tanto un entorno de desarrollo como un entorno de ejecución para la realización y mantenimiento de sistemas multiagente [107]. El entorno de desarrollo está formado por una serie de bibliotecas en Java que permiten la implementación de agentes de manera limpia e independiente de la plataforma sobre la que se va a ejecutar. El entorno de ejecución permite a los agentes vivir y comunicarse entre ellos. Está realizado enteramente en Java y proporciona una serie de herramientas que permiten al desarrollador controlar y depurar a los agentes en tiempo real [107]. Los aspectos más importantes en que se basa la plataforma JADE tenemos:

- Ejecución de agentes completamente asíncrona.
- Comunicación entre agentes en la misma o diferentes plataformas JADE/ LEAP [10], [108].
- Programación de agentes mediante un conjunto de paquetes Java.
- Validación de la ejecución mediante seguimiento mensajes y estado interno del agente.
- Es la plataforma más extendida porque implementa el estándar FIPA (FIPA, del inglés *Foundation for Intelligent Physical Agent*) [109].

Los agentes JADE necesitan del entorno de ejecución donde poder vivir. Cada instancia del entorno de ejecución se denomina contenedor (*container*). Al conjunto de los contenedores se le denomina plataforma (*platform*) y proporciona una capa que oculta a los agentes y al desarrollador, el entorno donde se ha decidido ejecutar la aplicación. Entre sus principales funciones y ventajas tenemos [107]:

- Ejecución distribuida.
- Interfaz gráfica para monitorización y depuración de agentes, incluyendo los que se encuentran en hosts remotos.
- Creación de agentes móviles.
- Ejecución de actividades en paralelo.
- Intercambio de mensajes ACL (ACL, del inglés *Agent Communications Language*) entre agentes.
- Registro automático de agentes (vía AMS).
- Servicio de nombres.

En cada plataforma debe existir un contenedor especial denominado contenedor principal (main container). La principal diferencia del contenedor principal respecto al resto de contenedores es que alberga dos agentes especiales:

- AMS proporciona el servicio de nombres asegurando que cada agente en la plataforma disponga de un nombre único. También representa la autoridad, es posible crear y matar agentes en contenedores remotos requiriéndoselo al agente AMS [110].

- *Directory Facilitator* (*DF*), proporciona el servicio de páginas amarillas. Gracias al agente DF, un agente puede encontrar otros agentes que provean los servicios necesarios para lograr sus objetivos [109].

Los agentes JADE tienen nombres únicos y se permite a cada agente descubrir a otros agentes y comunicarse con ellos mediante comunicaciones punto a punto. Los agentes proporcionan servicios, cada agente puede buscar a otros dependiendo de los servicios que proporcionen otros agentes.

La estructura de los mensajes se basa en el lenguaje ACL [110] que ha sido definido por la FIPA. Se ha escogido JADE como la herramienta de desarrollo, ya que cumple con las especificaciones FIPA y soporta la mayor parte de la infraestructura establecida como: protocolos de comunicación, codificación de mensajes y servicio de páginas amarillas.

La comunicación entre agentes se lleva a cabo a través de mensajes asíncronos, es decir, el agente que envía el mensaje y el destinatario del mensaje no tienen por qué estar disponibles al mismo tiempo. Es más, el destinatario no tiene porqué existir en ese instante. Los mensajes se pueden enviar a un agente en concreto o se pueden enviar a agentes que se desconocen pero se sabe que poseen unas ciertas características. La comunicación entre agentes es fundamental para poder conseguir la potencia propia de los sistemas multiagente. Para que los agentes se puedan comunicar deben usar el mismo lenguaje de comunicación. ACL permite transmitir una serie de conocimiento que viene expresado en un lenguaje de contenido. Toda la comunicación está basada en el intercambio de mensajes. La plataforma JADE que alberga al agente se encarga de hacerle llegar los mensajes a la plataforma del agente destinatario. La codificación decodificación de mensajes la hacen automáticamente los agentes.

JADE-LEAP

Los agentes escritos en JADE pueden ejecutarse en entornos móviles como teléfonos o asistentes digitales personales (PDA, del inglés *Personal Digital Assistant*) integrando las

redes inalámbricas junto con las redes convencionales. El problema es que JADE no puede correr en pequeños dispositivos debido a diversas razones:

- *Espacio*: la memoria necesaria para el entorno de ejecución es de varios megas.
- *Versión del JDK*: JADE necesita el JDK 1.4 o posterior, mientras que la mayoría de los dispositivos solo soportan *Personal Java* (*PJava*).
- *Características propias de las redes inalámbricas*: IP dinámicas, conectividad intermitente o bajo ancho de banda, hace que JADE no sea lo apropiado [10].

Para ello surge LEAP que permite ejecutar agentes JADE en dispositivos inalámbricos y/o conectados a través de redes inalámbricas [10] [111]. Los container del entorno de ejecución de JADE-LEAP se dividen en dos partes, un FrontEnd, que se ejecuta en el dispositivo móvil, y un BackEnd que se ejecuta en un servidor de la red fija [111].

2.4. Trabajos relacionados y estado del arte

Se han utilizado proxies para adaptar la velocidad de transmisión, teniendo en cuenta el estado del canal inalámbrico, en [112] se presenta una técnica de distribución de video que gestiona de forma inteligente el uso del ancho de banda y la capacidad de almacenamiento disponible en los servidores proxy, a través de la pre captura de una determinada cantidad de datos de video y almacenarlos a priori en los servidores proxy. El servidor proxy se utiliza para superar las restricciones al recuperar flujos multimedia a través de WAN, al reducir el requisito de ancho de banda del backbone.

Otros trabajos utilizan los proxies para solventar el problema de pérdida de paquetes al transmitir video streaming basados en los recursos previamente almacenados: en [65] se propone una técnica de almacenamiento en cache para mitigar los efectos de la latencia, perdida de paquetes y demoras, que despliega proxies multimedia a lo largo del camino del servidor al cliente. En esta técnica un servidor proxy almacena los flujos iniciales de un video largo; cuando en el futuro el cliente solicita el

flujo, el servidor proxy envía los flujos iniciales, además se conecta con el servidor del video y pide la retransmisión de los flujos posteriores. Se enfoca en garantizar la calidad del servicio al reducir el retraso y controlar los límites de rendimiento así como mejorar la continuidad del video streaming frente a un retraso o pérdida de paquetes en la red.

Una arquitectura de distribución de video asistida por un proxy, que reduce los requisitos exigidos al servidor y mejora la experiencia en el cliente se indica en [64]. Se desarrolla para ello, dos técnicas de video streaming asistidas por el proxy: captura asistida por el proxy y captura selectiva asistida por el proxy, las cuales proveen servicio instantáneo a un gran número de clientes. Esto incrementa la funcionalidad de la propuesta al implantar técnicas de selección de videos populares y asignación inteligente de recursos entre el servidor proxy y el servidor central.

Para proporcionar un servicio de tolerancia a interrupciones a los dispositivos inalámbricos, propone [113] el uso de un middleware en ambientes móviles basados en la intensidad de la señal de la red, *Signal Strength (SS),* aplicando un esquema de memorización intermedia (buffering) desarrollado en Java y aplicado a dispositivos inalámbricos Android.

Una manera de localizar zonas de interrupción propone [114] mediante un sistema para construir una base de datos basada en RSSI, que permite ubicar zonas de interrupción en locales interiores mediante la comparación de los valores RSSI medidos con los valores que se encuentran en la base de datos. Los valores que constituyen la base de datos son previamente tomados mediante un dispositivo LiDAR (LiDAR, del inglés *Laser Imaging Detection and Ranging*) y un receptor WiFi.

Para reducir la tasa de distorsión del video streaming [115], utiliza un proxy intermediario entre el servidor y el cliente, éste se encuentra localizado en el último salto de la red hacia el cliente, el cual coordina la comunicación usando un emisor de video streaming híbrido en un framework de optimización, dicho framework permite al proxy determinar que paquetes deben ser enviados desde el servidor de medios o retransmitidos directamente desde el proxy.

Un middleware de agente móvil que maneja el aprovisionamiento del contenido multimedia a los dispositivos clientes en redes inalámbricas plantea [116], esta solución se enfoca en predecir la movilidad del cliente entre áreas de cobertura, permitiendo la migración de los sistemas multiagente de forma personalizada, mediante la adaptación de los contenidos multimedia de acuerdo a los perfiles y preferencias de los usuarios y la anticipación de sus movimientos. Esta solución de predicción se caracteriza por ser ligera y descentralizada explotando la información obtenida del monitoreo del cliente acerca de la intensidad de la señal recibida de las estaciones base IEEE 802.11; las sesiones personalizadas de readecuación proactiva permiten mantener la continuidad del servicio.

Una alternativa de uso de los proxies como agentes de software de una plataforma propietaria para resolver las interrupciones se indica en [117], mantiene la continuidad del servicio de video streaming, cuando el cliente se mueve de una localidad inalámbrica a otra en tiempo de ejecución, generando un proceso de itinerancia. Para ello es importante la predicción de la entrega que permite migrar los proxies del móvil hacia redes inalámbricas donde el cliente móvil se reconectará y mediante el uso de un esquema de buffer proactivo y adaptativo se precargan los contenidos de multimedia.

Un esquema de localización colaborativo basada en los valores RSSI y filtrando aquellos que son afectados notablemente por las condiciones del entorno describe [118], para obtener una mejor aproximación de la localización de los dispositivos mediante el algoritmo del camino más corto y ubicar aquellos que tengan el mejor RSSI para compartir información de zonas de cobertura.

Para distribuir video bajo demanda [119], presenta un middleware basado en agentes, estos agentes se comportan como proxies, capaces de negociar el nivel de calidad y el flujo de flujos dependiendo de los perfiles y características del dispositivo y de las preferencias de usuario. Los componentes de esta propuesta pueden operar de forma autónoma para responder a los requisitos incluso en caso de la interrupción temporal del dispositivo.

Un sistema que permite la localización en interiores basada en WiFi, utilizando herramientas para visualización 3D como Google Earth, archivos KML e integrándolas en

un entorno Web se presenta en [120] para despliegue de las zonas de cobertura de los interiores.

Un mecanismo de sincronización multimedia, que reproduce los flujos en un entorno móvil de manera controlada y suave presenta [121], basa su mecanismo en un método de almacenamiento en buffer y un control efectivo de la reproducción del flujo, minimizando así el efecto de las característica de la red inalámbrica. Para implementar este entorno utilizan el concepto de proxy que negocia cómo van los flujos desde el buffer del servidor hacia los usuarios móviles. Este método propuesto mantiene la reproducción continua de los flujos y mantiene una presentación multimedia natural mediante el ajuste de la duración de la trama en pantalla.

Se plantea una solución middleware basada en proxies móviles que funcionan en los bordes de la red inalámbrica alámbrica [122], apoyando a los servicios continuos, sobre todo por los contenidos multimedia en la búsqueda de evitar interrupciones de streaming. Este mecanismo intenta explotar la migración de los proxies móviles con antelación a las celdas inalámbricas donde los clientes móviles van a volver a conectarse.

Una estrategia de buferización en dos niveles es planteada por [123], para mantener la continuidad del servicio de video streaming independiente de la movilidad del cliente para dispositivos inalámbricos.

El diseño de *Persistent Connectivity Management Protocol* (*PCMP*) para el proyecto *Drive-thru* se presenta en [124], el cual consta de un cliente y un proxy, para gestión de las pérdidas de paquetes debido a errores de bits en redes inalámbricas, lo que produce un impacto en el rendimiento de TCP. El PCMP se encarga de mantener únicamente las sesiones TCP dentro de la red inalámbrica que pasen a través del proxy.

Se discute en [125] por el desarrollo de un software multiagente para sistemas P2P video streaming de video basado en agentes. Presentan un agente software que mejorar la continuidad en sistemas de video streaming P2P, viendo el usuario en mejor calidad y menor retraso de extremo a extremo.

Otra solución tolerante a interrupciones y retrasos para compartir archivos mediante P2P en una MANET (MANET, del inglés *Mobile Ad Hoc Networks*) se plantea en [126]. Se implementa un modelo de comunicación asíncrona en el cual nodos en la red pueden delegar tareas a otros nodos y esperar por su respuesta, mejorando así el rendimiento en la transmisión de datos.

Una propuesta para reducir las interrupciones que se originan cuando un dispositivo móvil cambia de un medio inalámbrico a otro se indica en [127]. Enfocándose en el caso de cambiar desde un medio unicast a uno broadcast. Su solución propuesta reduce las interrupciones priorizando los paquetes de entrada justo antes de cambiar a broadcast.

Se plantean una solución de buferización en [128], para brindar la posibilidad de monitorear remotamente la actividad cardiaca de deportistas durante una carrera utilizando estaciones base ubicada a lo largo del camino. Los sensores transmiten los datos a la estación base; cuando se pierde la conexión, los datos son almacenados localmente en cache esperando tener nuevamente una conexión para enviarlos.

Existen trabajos, desarrollados por los integrantes de nuestro grupo de investigación, quienes presentan propuestas que posibilitan la continuidad del video streaming cuando ocurra una interrupción, estos los analizamos a continuación:

Se propone un nuevo protocolo ligero y de administración del buffer, para soportar eficientemente las interrupciones y recuperar automáticamente la sesión de video streaming en [129]. Se implementa a través de un cliente proxy en un dispositivo móvil y un servidor proxy para el control de las interrupciones. En caso de presentarse interrupciones el usuario es notificado y se le solicita que se mueva a un área de mejor cobertura, pero carece de reconexión automática ante las interrupciones.

Un software desarrollado con la técnica cross-layer, que no solo censa la congestión en redes WiFi, sino también el nivel de cobertura, la tasa de paquetes recibidos y perdidos para mejorar la QoS en aplicaciones de VoD (VoD, del inglés *Video on Demand*) con formato MPEG4 usando los protocolos RTSP/RTP se presenta en [130].

Otro mecanismo de control basado en la gestión de sesiones y de buffers proactivos de datos multimedia se presenta en [131], este utiliza agentes de software de la plataforma JADE-LEAP para gestionar las interrupciones presentadas en redes inalámbricas.

Una arquitectura basada en proxies y agentes inteligentes para dispositivos inalámbricos Android se indica en [132]. Permite restablecer automáticamente la sesión de video streaming, sin pérdida de información, generada por interrupciones en ambientes inalámbricos, pero se limita a un sistema operativo y disponer de la aplicación en cada dispositivo inalámbrico.

Un protocolo basado en la arquitectura de proxies es planteado en [133]. Este implementa la técnica de gestión de buffer para controlar interrupciones en la comunicación inalámbrica multimedia y recuperar la sesión de video streaming de forma automática, este mecanismo se aplicó a dispositivos inalámbricos.

Finalmente se propuso un mecanismo de control basado en a la gestión de buffers proactivos de datos multimedia manejados por agentes JADE LEAP en [134], Este se encargan de resolver la interrupción intermitentes WiFi y lograr la reanudación automática de las sesiones RTSP en dispositivos inalámbricos, limitados a la gama de celulares Nokia.

Analizados los trabajos, se puede concluir que han trabajado sobre plataformas específicas por la arquitectura heterogénea que presentan los teléfonos inteligentes, lo que genera implantar soluciones únicas y no generales. El uso de websockets no es posible porque éstos no manejan conexiones de datos usando RTP, solo comandos de control mediante el protocolo TCP. Los agentes inteligentes desarrollados solo establecen una comunicación sencilla entre ellos mediante el protocolo MTP.

Ninguno ha realizado un modelo basado en patrones software de diseño que permita generalizar el problema de servicio video streaming. Hasta donde alcanza nuestro conocimiento, nunca antes habían planteado modelos basados en patrones software de diseño para controlar interrupciones en servicios de video streaming y videoconferencia.

2.5. Contribuciones a la mitigación de las interrupciones de video

El servicio de video streaming ha llegado a ser muy utilizado, en especial por los usuarios con dispositivos inalámbricos. Esta creciente demanda del servicio de video streaming ha tomado fuerza por el uso de redes inalámbricas para consumir este servicio. Existe un serio problema al utilizar las redes inalámbricas como medio de transmisión, debido a su comportamiento impredecible que genera disrupciones en la comunicación. Nuestro interés es analizar cómo afecta éste problema al servicio desde la percepción del usuario.

2.5.1. Patrones software

Los patrones software de diseño facilitan la reutilización de una arquitectura software ya que proponen una solución genérica a un determinado problema. Cada patrón describe un problema que ocurre constantemente, entonces se representa la solución base al problema para que pueda ser utilizado repetitivamente sin hacer dos o más veces la misma tarea. Los patrones software de diseño son soluciones genéricas a problemas concretos que surgen en el desarrollo de software. Estos ayudan a resolver problemas de diseño que repetidamente ocurren, se enfocan en documentar y transmitir la experiencia en la solución al problema promulgando la reutilización.

En [135] indica que un buen patrón debe cumplir con las siguientes características:

- Soluciona un problema en un contexto en particular.
- Es recurrente lo cual permite dar una solución relevante a otras situaciones.
- Es adaptable a otros problemas dentro del mismo contexto.
- Son la base de una manual de ingeniería de software.
- Son conceptos probados, no teorías.

Se describe los patrones software utilizados en el presente trabajo en base a [136]:

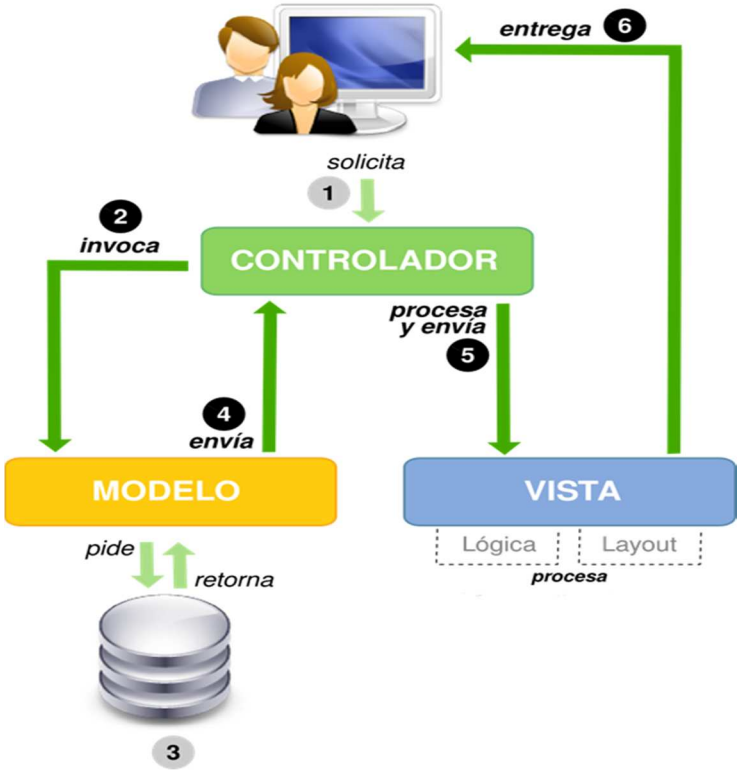


Figura 2.8: Patrón MVC

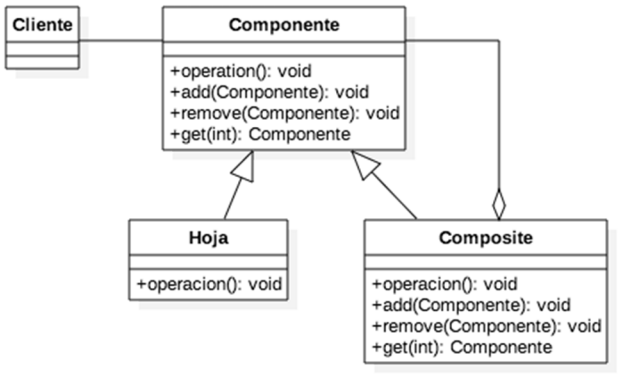


Figura 2.9: Patrón Composite

- *Patrón Modelo Vista Controlador* (*MVC*): es un patrón de diseño de software probado y que ayuda a entender y desarrollar mejores proyectos independientemente del lenguaje separando la parte de la vista (diseño de formularios, logos, colores, entre otros), la parte del modelo que se encarga

de la estructura de datos (base de datos) y el controlador, el cual se encarga de mantener la comunicación entre las anteriores y respondiendo a las peticiones de usuarios (Figura 2.8).

- *Patrón Composite*: este patrón permite construir objetos complejos partiendo de otros más sencillos lo que hace que se convierta en una estrategia recursiva, cuenta con elementos como (Figura 2.9):
  - *Componente:* en este se declara la interfaz para los objetos de la composición.
  - *Cliente:* se encarga de utilizar los objetos a través de la interfaz proporcionada por el componente.
  - *Composite:* en este se definen el comportamiento e implementa las operaciones del manejo de componentes.
  - *Hoja:* define comportamientos para objetos primitivos.

  El objetivo de este patrón es la facilidad de uso y apoya a la vista del patrón MVC.

- *Patrón Strategy*: facilita la implementación de varios comportamientos en clases hijas a través de una clase común, es decir, que en tiempo de ejecución de acuerdo al comportamiento dado en la aplicación se ejecutara una estrategia específica (Figura 2.10). Está compuesto por los componentes siguientes:
  - *Interfaz Strategy:* define el nombre del método que conforma la estrategia.
  - *Clases Strategy:* son aquellas que implementan la interfaz *Strategy*.
  - *Contexto:* en este componente se desarrolla la estrategia.

  El objetivo de este patrón es apoyar a la capa controlador en el patrón MVC con mayor nivel de fiabilidad y flexibilidad.
- *Patrón Observer*: se lo conoce como un patrón de comportamiento el cual relaciona diferentes objetos entre sí en torno a un objeto principal, es así que cuando el objeto principal cambia de estado los demás también cambien de forma automática (Figura 2.11). El objetivo de este patrón es mantener bajo acoplamiento y apoyar al modelo en el patrón MVC.

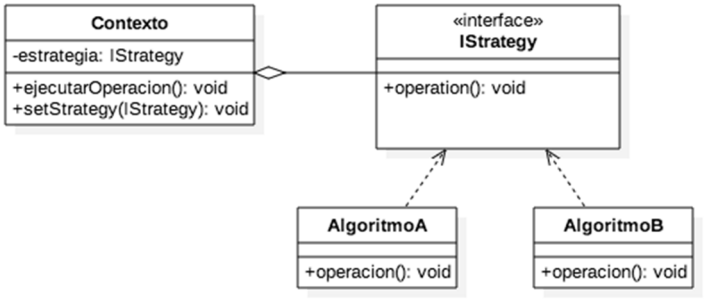


Figura 2.10: Patrón Strategy

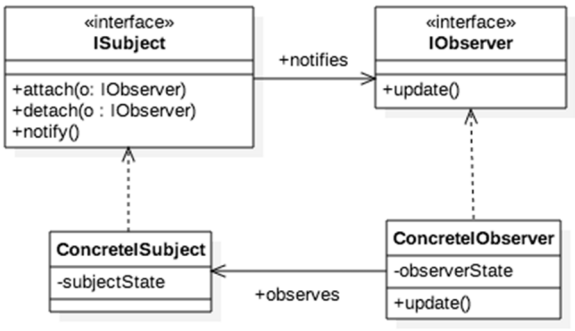


Figura 2.11: Patrón Observer

- *Patrón Proxy*: consiste en interponer un intermediario entre un objeto y los demás que lo utilizan (Figura 2.12). Presenta varios tipos: remoto, virtual y de protección. El *Proxy* remoto se encarga de la comunicación entre el cliente y el objeto remoto. El *Proxy* virtual coordina la instanciación de objetos que realizan operaciones computacionales costosas únicamente cuando el acceso al objeto es requerido. El *Proxy* de protección controla el acceso a un objeto en base a reglas de autorización.
- *Patrón Adapter*: permite que clases con distintas interfaces trabajen juntas, es decir, un objeto adaptador reenvía a otro objeto los datos que recibe a través de la implementación de métodos distintos. En la Figura 2.13 se observa la capa intermedia que comunica a dos clases con interfaces distintas, convirtiendo una estructura en otra, pero sin modificar la clase original.

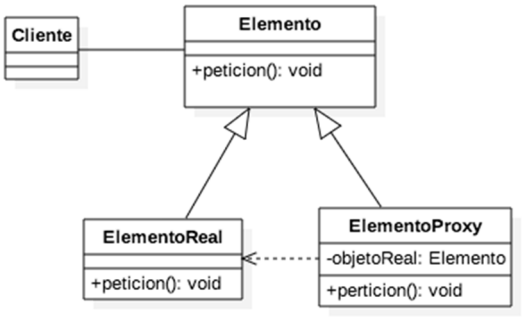


Figura 2.12: Patrón Proxy

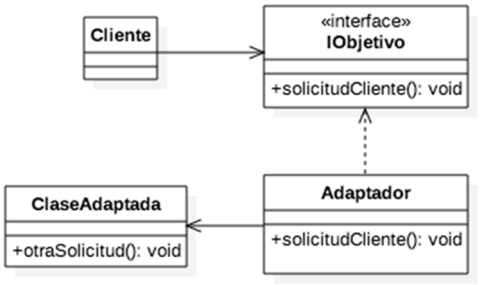


Figura 2.13: Patrón Adapter

2.5.2. Herramientas para servicio video streaming y videoconferencia

En este apartado explicamos las herramientas que nos permiten realizar superposición en tiempo real de imágenes, uso de reglas de negocio para toma de decisiones y representación de datos geográficos en 3D y video en tiempo real en navegadores.

Realidad Aumentada

La *Realidad Aumentada* (*RA*) es una tecnología que trata de incluir información generada por cualquier dispositivo informático sobre el entorno del mundo real, para así permitir la interacción del usuario con objetos virtuales [137].

Permite enfocar con un dispositivo informático y visualizar por pantalla una interpretación del Mundo real, diferente a lo que se puede visualizar con nuestro propio sentido de la vista. RA puede ser implementada de dos maneras:

- *Basada en posición*: muestra información enriquecida del entorno, apoyada en la posición y orientación.
- *Basada en marcadores*: se fundamenta en reconocer una imagen (patrón) y superponer otras imágenes en base a la posición y orientación del mismo, simulando que forma parte del mundo real.

Una aplicación con RA debe cumplir con ciertas características principales y fundamentales, con respecto a su funcionalidad y estas son:

- Interacción entre el mundo real y los objetos virtuales, mediante la utilización de las 3 dimensiones proyectadas por la cámara del dispositivo.
- Ejecución en tiempo real.

Las configuraciones de hardware en RA se basan en los equipos de sobremesa estática con cámaras fijas o en cascos de realidad virtual (HMD, del inglés *Head Mounted Display*) con ordenadores portátiles incorporados.

Cuando se piensa en movilidad, los dispositivos inalámbricos de mano aparecen como soluciones válidas para las aplicaciones de RA móvil, estas tecnologías son más accesibles al público por sus bajos costos [138].

Utilizamos el framework *Look!* [139] para el desarrollo de RA en plataformas Android, cuya principal ventaja es que su código es abierto (LGPL v3) y cuenta con capacidad de extensión y mejoramiento continuo en base a las necesidades que se presenten, en lo que respecta a: RA, localización en interiores, integración con servicios remotos y servicio de persistencia de datos.

Look!, además de las funcionalidades que nos ofrece para RA incorpora una parte fundamental para lo que es la geolocalización, dando así paso a una estructura modular.

Reglas de negocio

Dentro del contexto de aplicaciones empresariales, existe el concepto de regla de negocio. Estas reglas son definidas por las directivas de la organización y pueden ser condiciones o parámetros de los diferentes servicios que ésta presta [140]. Algunos ejemplos son:

- El precio de un minuto de telefonía celular, según el plan al que pertenezca el usuario.
- Las condiciones para aceptar o rechazar una solicitud de crédito.
- Los parámetros para realizar descuentos por compra de productos en combo.
- Las condiciones para admitir a un estudiante en una Universidad.

Las reglas evolucionan a lo largo del ciclo de vida de la organización debido a su estrecha dependencia de los motivadores de negocio que puede tener una organización y las fuerzas externas. Por tal razón, el tiempo de respuesta ante dicha evolución debe ser el mínimo posible al igual que el impacto económico ante un cambio en un motivador o en una fuerza externa. Es así como la decisión de mantener dentro del código de una o varias aplicaciones de la empresa las reglas de negocio, tiene gran impacto en especial económico. Específicamente debido a la cantidad de cambios que se puedan requerir, para ajustar el código en las aplicaciones en el momento en que apremia satisfacer una necesidad de negocio basada en una nueva regla o en el cambio de una de éstas.

Los motores de reglas de negocio o BRMS (BRMS, del inglés *Business Rule Management System*) surgen como una alternativa de solución a la problemática de administrar el cambio de las reglas de negocio en una organización, en nuestro caso con agentes de software [9]. En particular los BRMS ofrecen:

- Un repositorio común a las aplicaciones donde se guardan las reglas de negocio versionadas.
- Herramientas que permiten definir estas reglas tanto a usuarios técnicos (desarrolladores) como a usuarios no técnicos (directivos, expertos de negocio).

- Independencia entre el lenguaje de programación de una aplicación y el lenguaje para expresar las reglas.
- Facilidad para definir las reglas de negocio, por categorías, en un lenguaje de alto nivel propio del motor de reglas.
- Un mecanismo de despliegue de las reglas de negocio.

Lenguaje de marcado basado en XML para representación de datos geográficos en 3D

Un archivo en formato KML [141] contiene coordenadas, direcciones, altura, entre otras variables que permiten representar en un mapa, una ruta o punto de interés, en otras palabras, es el lenguaje de marcado basado en XML (XML, del inglés *eXtensible Markup Language*) para representar datos geográficos en tres dimensiones [142]. Es reconocido como un estándar en el *Open Geospatial Consortium* [143]. Un archivo KML especifica una característica (un lugar, una imagen, una posición, una ruta o un polígono), que contienen coordenadas (latitud y longitud), que en muchas ocasiones puede venir comprimido en un archivo KMZ (KMZ, del inglés *Keyhole Markup Language Zip*) [144]. Un archivo KML tiene la siguiente estructura:

1. Cabecera del XML:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   ```

2. Espacio de nombres o Namespace propio de KML:

   ```
   <kml xmlns="http://www.opengis.net/kml/2.2"
   xmlns:gx="http://www.google.com/kml/ext/2.2"
   xmlns:kml="http://www.opengis.net/kml/2.2"
   xmlns:atom="http://www.w3.org/2005/Atom">
   ```

3. El objeto Placemark que hace referencia a la posición, dividido en nombre, descripción y conjunto de coordenadas que conforman el camino:

   ```
   <Placemark>
   <name>Ruta de Prueba</name>
   <styleUrl>#m_ylw-pushpin</styleUrl>
   <LineString>
   ```

<tessellate>1</tessellate>
<coordinates>
-78.45876922394318,-0.2907473038582219,0 -78.45962845232083,-0.2910824364469906,0 -78.45926872660962,-0.2918885432418332,0 78.45885131304623,-0.2924142379706846,0 -78.45776061343645,-0.2945232380020063,0 -78.45716919796966,-0.2955813292945207,0 78.45587564927628,-0.297815041302652,0 -78.45679124311162,-0.2983939244213915,0 -78.45828346321,-0.2991575516721163,0
</coordinates>
</LineString>
</Placemark>

Un archivo KML para dispositivos inalámbricos permite los siguientes elementos:

- Marcas de posición con elementos de nombre <*name*>.
- Puntos, íconos, carpetas.
- Elementos HTML.
- KMZ (KML comprimido que incluye imágenes adjuntas).
- Cadenas de línea y polígonos.
- Aspectos como el color, el relleno y la opacidad.

Para crear un archivo KML se puede recurrir desde un editor de texto plano, como un programa especializado de coordenadas que nos genere este archivo, como se presenta en la Figura 2.14.

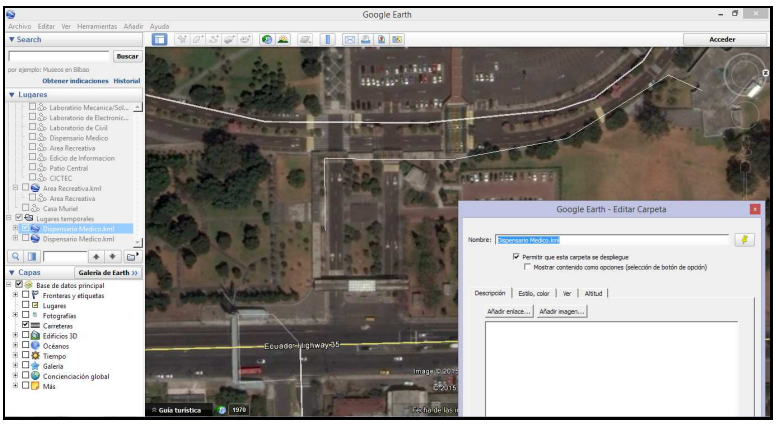


Figura 2.14: Archivo KML generado desde Google Earth

Los archivos KML sirven para la creación de modelos y el almacenamiento de coordenadas geográficas como puntos, rutas, imágenes, polígonos, entre otros; las cuales se pueden compartir o utilizar para referenciar un lugar o información específica con otros usuarios.

HTML5

HyperText Markup Language 5 (*HTML5*)) [145] corresponde a la quinta revisión del lenguaje básico de la Web. Establece nuevos elementos y atributos, introduce elementos multimedia como la etiqueta <video> y <audio> que permiten el despliegue en navegador mediante interfaces estándar.

La comunicación, se implementa mediante el uso de API (API, del inglés *Application Programming Interface*) WebSocket que permiten una comunicación bidireccional entre el servidor y el navegador Web.

La etiqueta <video> permite reproducir archivos de vídeo en la página Web sin necesidad de usar plugins o el uso de herramientas de terceros como Flash. La personalización del reproductor se lo realiza mediante el uso de hojas de estilo CSS (CSS, del inglés *Cascading Style Sheets*) y JavaScript, entregando una apariencia única al reproductor.

Entre los formatos aceptados por HTML5 para la etiqueta de video tenemos: *H.264* [48], conocido también como MPEG-4 parte 10; OGM (OGM, del inglés *Ogg Media*) [146] que utiliza como CODEC de vídeo a Theora y de audio a Vorbis; *WebM* [66], que soporta para audio el CODEC Vorbis, y para video el CODEC VP8.

Tabla 2.2: CODEC de audio y video soportados por navegadores Web

FORMATO	CHROME	FIREFOX	INTERNET EXPLORER	OPERA	SAFARI
WebM (VP8+Vorbis)	SI	SI	NO	SI	NO
OGG (Theora+Vorbis)	SI	SI	NO	SI	NO
MP4 (H.264+MP3)	SI	NO	SI	NO	SI
MP4 (H.264+AAC)	SI	NO	SI	NO	SI

En la Tabla 2.2 se muestra una lista de los CODEC de audio y video soportados por los navegadores más populares, donde se observa que al menos un formato de audio y vídeo es compatible para HTML5

Uno de los aportes importantes de HTML5 es el uso del protocolo WebSocket que establece un canal de comunicación bidireccional entre el navegador Web y el servidor sobre una única conexión TCP [147].

Está diseñado para funcionar sobre los puertos 80 y 443, además sobre proxies HTTP y no limita el uso de HTTP en Websocket. Permite mantener conexiones persistentes entre el navegador y el servidor para un intercambio de información independiente del tiempo [148]. Los navegadores en dispositivos inalámbricos que soportan el estándar HTML5 son Chrome, Firefox y Opera [146]. En conclusión HTML5 se caracteriza por presentar cambios significativos en tres aspectos: estructura, nuevos elementos y funcionalidad [145], [149].

Dynamic Adaptive Video streaming over HTTP

Dynamic Adaptive Streaming over HTTP (*DASH*) es una solución desarrollada por MPEG y publicada como ISO/IEC 23009-1; 2012 [150]. Presenta algunas ventajas indicadas en [150–153] y que las podemos resumir:

- Permite el uso de conexiones HTTP/TCP para reutilizar la infraestructura de red existente.
- Brinda disponibilidad de recursos a gran escala de los servidores basados en RTP.
- Mejora la QoE mediante la entrega de servicios video streaming adaptativos [154].

En la Figura 2.15 se observa el escenario donde el cliente provee al sistema de cierto grado de adaptación requiriendo los segmentos que más se adecúen al ancho de banda disponible [155]. Estos segmentos se codifican en múltiples versiones, cada uno con diferentes parámetros de codificación, por lo tanto se dispone de múltiples niveles de

calidad con el mismo contenido multimedia. El cliente puede solicitar estos segmentos para su despliegue mediante la URL asignada a cada uno.

DASH principalmente define dos formatos:

- *Media Presentation Description* (*MPD*) [156]: que es un archivo de metadatos que describe el contenido disponible y algunas otras características.
- Formato de los segmentos.

El cliente de DASH debe seguir el siguiente proceso para reproducir el contenido multimedia [152]:

- Obtiene el archivo MPD mediante HTTP, correo electrónico u otro tipo de transporte.
- Extraer la información necesaria para la reproducción.
- Obtenida la información, selecciona las características adecuadas y empieza a realizar la transmisión multimedia mediante peticiones HTTP de los segmentos que cumplen con las características seleccionadas.

MPD es un documento XML con el que un cliente DASH obtiene los metadatos necesarios para acceder a los segmentos y proporcionar un servicio de streaming al usuario. La estructura jerárquica de MPD, conformada por:

- *Period*: determina el intervalo temporal del contenido multimedia y puede contener uno o varios *adaptation sets*.
- *Adaptation set*: proporciona información de las diferentes versiones de codificación de los contenidos (bitrate), lo que implica tener diferentes *representations*.
- *Representation*: es la codificación a ser entregada y puede tener uno o varios *segments*.
- *Segment*: son cada uno de las partes en las que se encuentra dividido los contenidos y contienen su propia URL para que el cliente puede descargarse mediante peticiones GET HTTP.

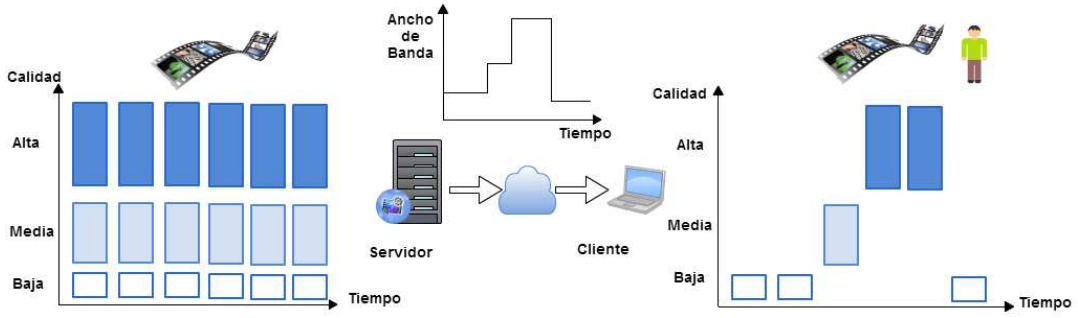


Figura 2.15: Escenario de adaptación DASH

Un sistema de distribución multimedia, cuya arquitectura básica está basada en la tecnología DASH, dispone de un servidor HTTP y un cliente. Se puede añadir elementos intermedios como: cache HTTP y proxy, con el propósito de mejorar la eficiencia del servicio. La descripción de cada elemento, la presentamos a continuación:

- *Servidor HTTP*: se encarga de almacenar archivos MPD asociados con la descripción de los medios y los contenidos codificados siguiendo el estándar DASH. Es un servidor HTTP convencional.
- *Caché HTTP*: es un espacio de almacenamiento para peticiones y respuestas HTTP. Si llega una petición HTTP similar a las almacenadas, envía la respuesta sin tener que consultar al servidor.
- *Cliente*: se encarga de pedir y reproducir los contenidos al servidor Web mediante peticiones GET estándar. El control y la lógica de adaptación al ancho de banda disponible reside en el lado del cliente, dando solución a problemas de escalabilidad en el lado del servidor.

Es importante indicar que el estándar DASH solo describe a MPD y el formato de los segmentos. La forma en la que se realiza el envío de MPD, la codificación de los medios y el comportamiento del cliente sobre el proceso de estimación del ancho de banda y las decisiones acerca de la solicitud de los segmentos para llevar a cabo la adaptación de los contenidos no se encuentran definidos en el estándar DASH [152].

La fortaleza de la tecnología DASH está en el uso de protocolos ampliamente manejados para el intercambio de información en Internet, que no eran considerados

factibles para efectuar una transmisión de contenido multimedia. El uso de DASH permite la utilización de este tipo de protocolos, en escenarios video streaming, proveyendo de cierto grado de adaptación en función de las capacidades del cliente y del estado de la red. El video streaming adaptativo sobre HTTP esté siendo gradualmente adoptado por los proveedores de contenidos y servicios en la red, gracias a las ventajas brindadas en términos de uso de recursos y calidad percibida por el usuario.

WebRTC

WebRTC (WebRTC, del inglés *Web Real-Time* Communications) es un proyecto de software libre mantenido por Google, Mozilla y Opera y es parte de la propuesta HTML5. Se encuentra definida entre el *World Wide Web Consortium* (*W3C*) y el *Internet Engineering Task Force* (IETF) y permite habilitar las capacidades *Real Time Communication* (*RTC*) entre navegadores de internet usando simplemente API JavaScript, proporcionando audio, video y datos P2P (P2P, del inglés *Peer-to-Peer*)[157] sin necesidad de plugins [158]. En el experimento realizado en el capítulo 5 se lo utilizó por su característica de ser gratuito y de código abierto.

WebRTC implementa las Web API siguientes [87]:

- *MediaStream*: permite acceder a los flujos de media del equipo, tales como cámara, micrófono u otro dispositivo de captura. Esta petición se realiza a través de la función *getUserMedia*.
- *RTCPeerConnection*: establece la conexión entre pares (navegador a navegador), para transmitir la información.
- *RTCDataChannel*: permite la comunicación de otros tipos de datos en tiempo real, tales como chat de texto, transferencia de archivos…

WebRTC no implementa la gestión de sesiones o también llamada señalización abstracta, lo deja a JSEP (JSEP, del inglés *Java Session Establishment Protocol*) [159], quien se encarga de controlar el mecanismo de señalización a través de JavaScript, este

mecanismo elimina casi por completo al navegador del núcleo de flujo de señalización, la única interfaz que necesita es que la aplicación envié las descripciones de sesión e interactúe con ICE (ICE, del inglés *Interactive Connection Establishment*) [160].

Como protocolos de transporte WebRTC utiliza a UDP para la transmisión de audio y video en tiempo real; SRTP (SRTP, del inglés *Secure Real-Time Transport Protocol*) para proveer seguridad y protección al transporte de recursos RTP; RTCP para supervisar las estadísticas de transmisión de flujos de datos.

Para dar solución al NAT (NAT, del inglés *Network Address Translation*) [161], WebRTC presenta tres soluciones: STUN (STUN, del inglés *Session Traversal Utilities for NAT*) [162] protocolo de red tipo cliente-servidor utilizado por clientes NAT para encontrar su dirección IP pública; TURN (TURN, del inglés *Traversal Using Relay NAT*) [163] extensión del protocolo STUN, utilizado para retransmitir video o datos en una comunicación punto a punto; ICE se lo considera como un marco que define el uso de los protocolos STUN y TURN para poder atravesar NAT, mediante el tratamiento de distintas rutas para establecer la comunicación entre dos usuarios, sin la necesidad de emplear otros servicios [87].

WebRTC soporta diversos CODEC para audio y video, los más utilizados son OPUS y VP8 respectivamente; destacan por su bajo consumo de ancho de banda, baja latencia y soporte de hardware de cliente heterogéneo y fueron utilizados en el experimento porque son gratuitos y de código abierto [87]. WebRTC dispone de un agente que permite extraer métricas de comunicación en una aplicación web, para obtener estas estadísticas se han definido una serie de API JavaScript que permiten obtener información estadística acerca de una comunicación [164].

2.5.3. Esquema básico cliente servidor de video streaming con patrones

Se utiliza el modelo de caso de usos para describir la funcionalidad del servicio de video streaming. En la Figura 2.16 y tabla 2.3-2.4 se muestra la relación entre los actores y los casos de en este servicio

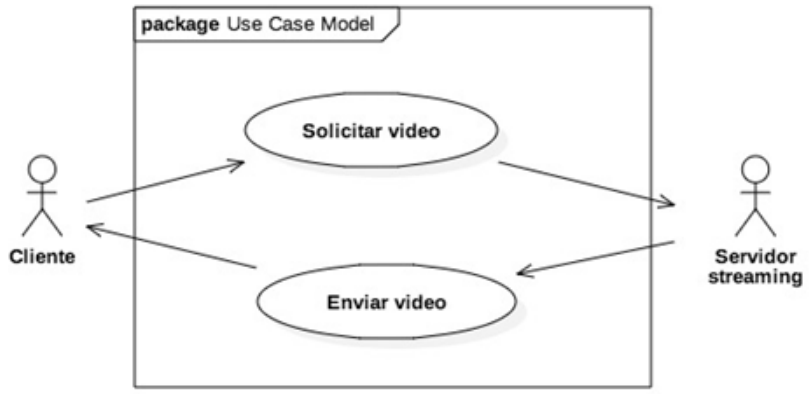


Figura 2.16: Casos de uso servicio video streaming

Tabla 2.3: Casos de uso servicio video streaming – solicitar video

Caso de uso: solicitar video
Actores: cliente, servidor video streaming
Resumen: describe como inicia la petición del video
Precondiciones:
Descripción: 1. El cliente solicita la visualización del video. 2. Ingresa los datos del protocolo, IP del servidor, puerto y video. 3. Recibe parámetros de sesión
Pos condiciones:
Observaciones:

Tabla 2.4: Casos de uso servicio video streaming - enviar video

Caso de uso: enviar video
Actores: servidor video streaming, cliente
Resumen: describe como realiza la entrega del video al cliente
Precondiciones:
Descripción: 1. En función de los datos con que solicito video toma la información de: 1.1 Librería de videos. 1.2 Dispositivo de captura de video. 2. Envía al cliente el video en flujos. 3. Cliente recibe flujo y lo despliega.
Pos condiciones:
Observaciones:

El primer caso de uso del servicio video streaming describe como inicia la petición de video, explicado en la Tabla 2.3.

El segundo caso de uso del servicio video streaming describe como se realiza la entrega del video al cliente, explicado en la Tabla 2.4.

Para el diseño del mecanismo mediante patrones software se utilizó el patrón MVC por presentar una arquitectura formada por tres niveles que permite separara la lógica del negocio y la presentación, permitiendo un desarrollo más sencillo de las aplicaciones.

Esto permite tener una mejor estructura y orden al momento de generar código, integridad de los datos y reutilización de componentes. El diseño del mecanismo para el servicio de video streaming se lo observa en la Figura 2.17.

Como se indicó MVC se compone de tres capas, en cada una de ellas se implementó lo siguiente:

- *Modelo*: ubica al almacenamiento donde se aloja la librería de videos del servidor video streaming.
- *Vista*: se encuentran las clases que permiten desplegar el servicio de video streaming al cliente, de acuerdo a los parámetros de la sesión establecida.
- *Controlador*: coordina la interacción entre el cliente y el servidor para la visualización del video.

Basado en el modelo creado mediante patrón MVC se establece el diagrama de clases que se muestra en la Figura 2.18. Cada una de estas clases se encuentra asociada una de las capas de MVC así:

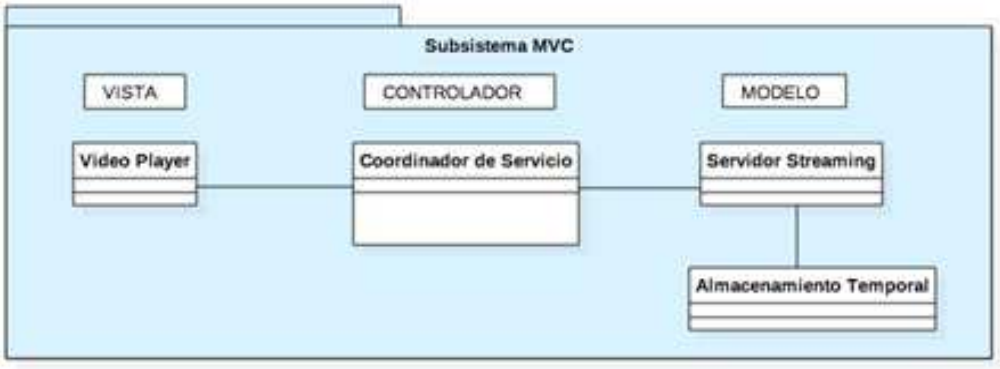


Figura 2.17: Modelo patrones software para servicio video streaming

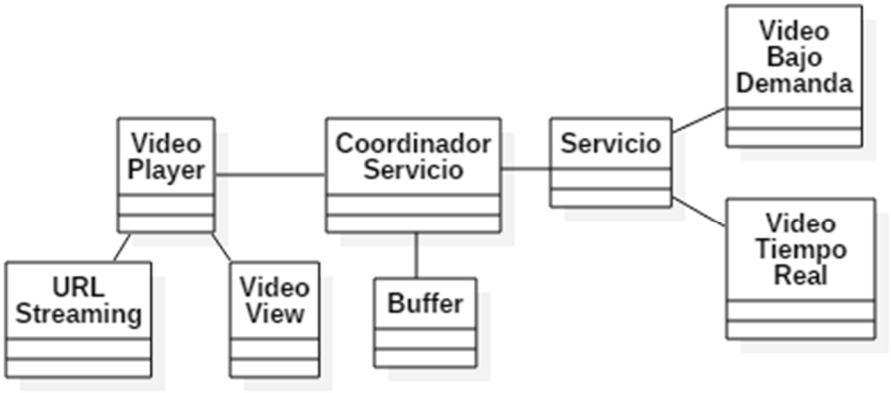


Figura 2.18: Diagrama de clases servicio video streaming

- *Modelo*: contiene las clases *Video Bajo Demanda*, *Video Tiempo Real* y *Buffer*.
- *Vista*: contiene *Video Player*, *Video View* y *URL Streaming*.
- *Controlador*: contiene *Coordinador Servicio* y *Servicio*.

En la Figura 2.19 se describe el diagrama de secuencia que indica la interacción entre el cliente y el servidor al solicitar la visualización de un video. El cliente solicita el video mediante la clase *Video Player*. Esta solicitud debe ir acompañada del parámetro definido en la clase *URL Streaming*. Con esta información se eleva la petición del servicio al *Coordinador de Servicio*. Éste pide a la clase *Servicio* indique los parámetros que describan la sesión para establecerla.

Establecida la sesión la clase *Servicio* solicita a una de las dos fuentes de video que dispone, de acuerdo al parámetro indicado en la clase *URL Streaming*. Las fuentes de video son las clases *Video Bajo Demanda* y *Video Tiempo Real*. Cada una de ellas se encuentra asociada a una librería de videos y un dispositivo de captura de video respectivamente. Una de las fuentes procede a entregar el video a la clase *Coordinador de Servicio* para que éste establezca el flujo de datos con la clase *Buffer* del cliente. Finalmente la clase *Buffer* recibe la información y la entrega a la clase *Video View* para su visualización.

La Figura 2.20 describe el diagrama de despliegue del servicio de video streaming. Contiene dos usuarios: cliente y servidor. El cliente corresponde a cualquier

dispositivo móvil y contiene las clases: *Video Player*, *Video View*, *URL Streaming* y *Buffer*. El servidor es un servidor de aplicaciones que contiene las clases: *Servicio*, *Video Bajo Demanda* y *Video Tiempo Real*. Se debe indicar que la clase *Coordinador de Servicio* de la capa controlador se encuentra distribuida entre el cliente y el servidor con el propósito de atender sus peticiones y respuestas.

Utilizando el diagrama de despliegue, explicado anteriormente, *Video Player* instancia mediante mensaje interno a *URL Streaming* para obtener parámetros. Con esta información *Video Player* envía una petición mediante protocolo RTSP pidiendo parámetros de sesión al servidor.

El servidor mediante protocolo SDP envía la descripción de la sesión al cliente. Establecida la sesión, el *Coordinador de Servicio* solicita mediante mensaje interno al Servicio enviar el video de la fuente indicada en *URL Streaming*. El servidor entrega el flujo de datos del video de la fuente seleccionada al *Coordinador de Servicio* mediante mensaje interno. El *Coordinador de Servicio* envía el flujo de datos mediante protocolo RTP al *Buffer* del cliente. El *Buffer* mediante mensaje interno entrega al *Video View* el flujo de datos para su visualización.

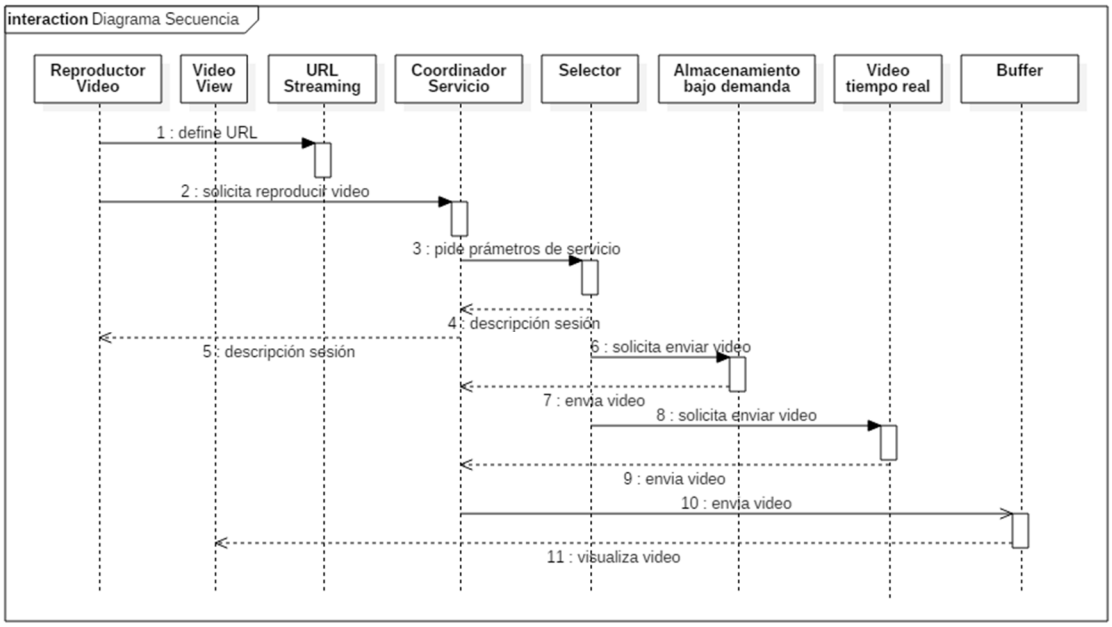


Figura 2.19: Diagrama de secuencia servicio video streaming

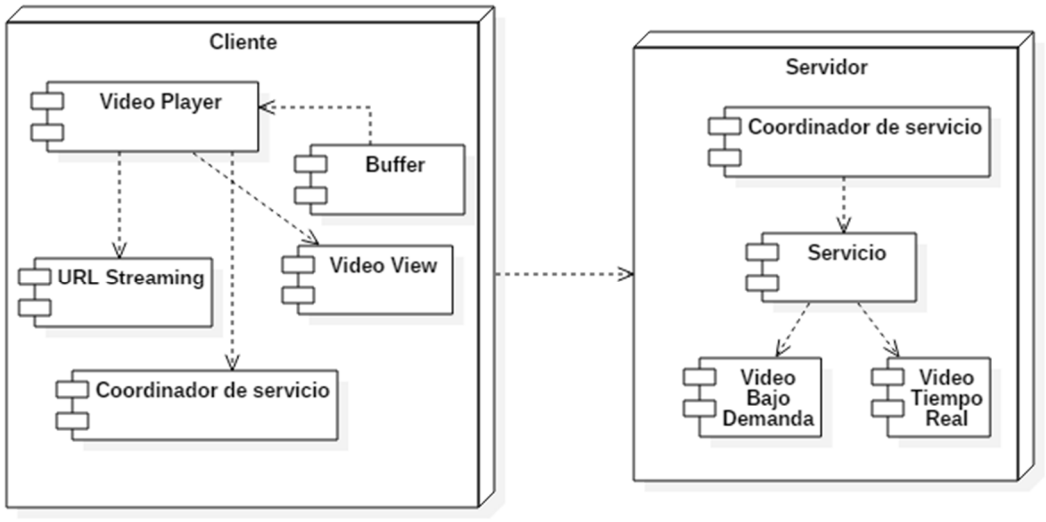


Figura 2.20: Diagrama de secuencia servicio video streaming

2.5.4. Modelo matemático del rendimiento del esquema básico

Es importante analizar el impacto de las interrupciones en la transmisión del video, para ello se ha seleccionado dos escenarios:

- *Peor escenario:* donde cada interrupción que ocurra se dará muy cercano a la finalización del video
- *Mejor escenario:* cada interrupción se dará en lo más cercano al inicio del video.

Análisis peor escenario

En la Figura 2.21 se puede observar el comportamiento de los flujos transmitidos con *n* interrupciones, en condiciones de peor escenario.

Si analizamos la Figura 2.21 podemos observar que por cada interrupción presentada, tenemos un flujo de datos que tiene un T_{FLUJO} que corresponde al tiempo de duración medido en segundos. Si observamos el último flujo transmitido T_{FLUJOn}, vemos que este se encuentra constituido por bloques de T_{DATO} desde T_{DATO1} hasta T_{DATOn}.

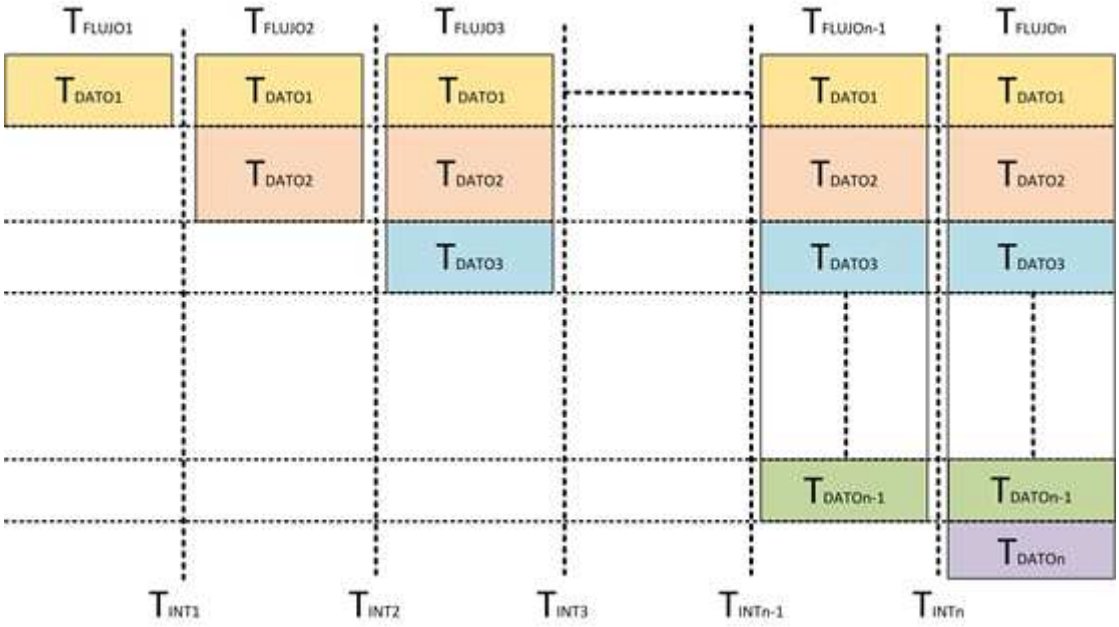


Figura 2.21: Comportamiento flujos peor escenario

Cada uno de estos T_{DATO} conforma el tiempo de retransmisión acumulado en el escenario, excepto el último T_{DATO} del último flujo que solo se transmite una vez.

Si ordenamos los flujos de menor a mayor duración, podemos realizar la resta entre el flujo con mayor duración (T_{FLUJOn}) y el flujo que le sigue en duración ($T_{FLUJOn-1}$); obteniendo como resultado el valor de (T_{DATOn}) en segundos, correspondiente al tiempo que el flujo (*n*) no retransmitió datos en esa interrupción (*n*).

$$T_{DATO_n} = T_{FLUJO_n} - T_{FLUJO_{n-1}} \quad (1)$$

De esta manera puedo obtener el tiempo de cada uno de los T_{DATO} último, de cada flujo generado por una interrupción y que conforman el T_{FLUJOn}.

Es importante observar que en este escenario solo el último T_{DATO} del último flujo transmitido no presenta retransmisión en todo el escenario analizado. Como las interrupciones se presentaron cercanas a la finalización del video streaming, se concluye en primera instancia que se trata de un escenario que presenta un elevado aporte de tiempo de retransmisión de T_{DATO} acumulado en todo el tiempo de ejecución del escenario.

Análisis mejor escenario

En la Figura 2.22 puede observarse el comportamiento de los flujos transmitidos frente a n interrupciones, en el mejor escenario.

Si analizamos la Figura 2.22 podemos observar que presenta similar comportamiento al escenario peor. Por cada interrupción presentada, tenemos un flujo de datos que tiene un T_{FLUJO} que corresponde al tiempo de duración medido en segundos. Si observamos el último flujo transmitido T_{FLUJOn}, vemos que este se encuentra constituido por bloques de T_{DATO} desde T_{DATO1} hasta T_{DATOn}. Cada uno de estos T_{DATO} conforma el tiempo de retransmisión acumulado en el escenario, excepto el último T_{DATO} del último flujo que solo se transmite una vez.

Si ordenamos los flujos de menor a mayor duración, podemos realizar la resta entre el flujo con mayor duración (T_{FLUJOn}) y el flujo que le sigue en duración (T_{FLUJO1}); obteniendo como resultado el valor de T_{DATO} (T_{DATOn}) en segundos, correspondiente al tiempo que el flujo (*n*) no retransmitió datos en esa interrupción (*n*).

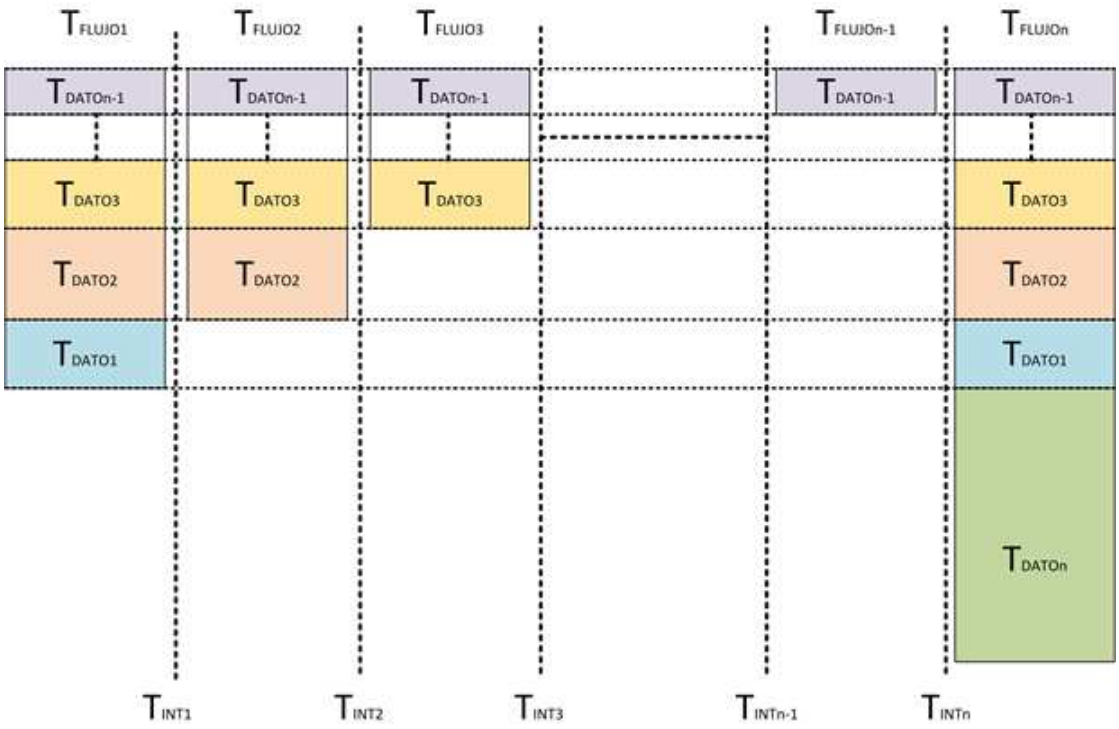


Figura 2.22: Comportamiento flujos mejor escenario

$$T_{DATO_n} = T_{FLUJO_n} - T_{FLUJO_1} \ (2)$$

De esta manera puedo obtener el tiempo de cada uno de los T_{DATO} último, de cada flujo generado por una interrupción y que conforman el T_{FLUJOn}.

Es importante observar que en este escenario solo el último T_{DATO} del último flujo transmitido no presenta retransmisión en todo el escenario analizado. Como las interrupciones se presentaron cercanas a la inicialización del video streaming, se concluye en primera instancia que se trata de un escenario que presenta un bajo aporte de tiempo de retransmisión de T_{DATO} acumulado en todo el tiempo de ejecución del escenario.

Este análisis aplica para cualquier escenario que se presente. Es necesario ordenarlos, en este caso ascendentemente, y así nos ajustamos a la Figura 2.21 de peor escenario, con lo cual podemos utilizar la fórmula descrita para calcular el tiempo de cada T_{DATO} último de cada flujo generado por una interrupción.

A continuación se explica el detalle de la obtención de la ecuación del tiempo total de ejecución del video en este modelo.

Si analizamos las Figuras 2.21 y 2.22 tenemos que el $T_{EJECUCION}$ está determinado por la siguiente ecuación:

$$\begin{aligned} T_{EJECUCION} = {} & T_{INICIO_{FLUJO1}} + T_{INICIO_{FLUJO2}} + T_{INICIO_{FLUJO3}} + \cdots + T_{INICIO_{FLUJOn}} \\ & + (n+1) \times T_{DATO1} + n \times T_{DATO2} + (n-1) \times T_{DATO3} + \cdots + 2 \\ & \times T_{DATOn-1} + T_{DATOn} + T_{INTERRUPCION1} + T_{INTERRUPCION2} \\ & + T_{INTERRUPCION3} + \cdots + + T_{INTERRUPCIONn} \end{aligned}$$

Donde:

$T_{INICIOFLUJO}$ es el tiempo de establecimiento de la sesión de video streaming.

T_{DATO} es el tiempo de transmisión de datos en el flujo.

$T_{INTERRUPCIÓN}$ es el tiempo de duración de la interrupción.

n es el número de interrupciones.

Asumiendo que el servicio de video streaming presenta iguales tiempos de establecimiento en todos los casos, entonces el $T_{INICIOFLUJO}$ presentan el mismo tiempo de duración, por lo que:

$$T_{INICIO_{FLUJO1}} = T_{INICIO_{FLUJO2}} = T_{INICIO_{FLUJO3}} = \cdots = T_{INICIO_{FLUJOn}} = T_{INICIO_{FLUJO}}$$

Entonces:

$$T_{EJECUCION} = (n+1) \times T_{INICIO_{FLUJO}} + (n+1) \times T_{DATO1} + n \times T_{DATO2} + (n-1) \times T_{DATO3} + \cdots + 2 \times T_{DATOn-1} + T_{DATOn} + T_{INTERRUPCION1} + T_{INTERRUPCION2} + T_{INTERRUPCION3} + \cdots + +T_{INTERRUPCIONn}$$

Luego si agrupamos los T_{DATO} transmitidos más de una vez, agrupamos las $T_{INTERRUPCION}$ presentadas y separamos el T_{DATOn} transmitido una sola vez, tenemos:

$$T_{EJECUCION} = (n+1) \times T_{INICIO_{FLUJO}} + \{(n+1) \times T_{DATO1} + n \times T_{DATO2} + (n-1) \times T_{DATO3} + \cdots + 2 \times T_{DATOn-1}\} + (T_{INTERRUPCION1} + T_{INTERRUPCION2} + T_{INTERRUPCION3} + \cdots + T_{INTERRUPCIONn}) + T_{DATOn}$$

Ahora si aplicamos sumatorio tenemos la ecuación que determina el $T_{EJECUCION}$ del modelo de servicio video streaming, nuestra ecuación 3:

$$T_{EJECUCION} = (n+1) \times T_{INICIO_{FLUJO}} + \sum_{i=1}^{n} \left((i+1) \times T_{DATO_{n+1-i}}\right) + \sum_{i=1}^{n} T_{INTERRUPCION_i} + T_{DATOn} \quad (3)$$

Donde:

$(n+1) \times T_{INICIO_{FLUJO}}$ es el tiempo total de inicio de los flujos transmitidos por n interrupciones.

$\sum_{i=1}^{n}\left((i+1)\times T_{DATO_{n+1-i}}\right)$ es el tiempo total de T_{DATO} transmitidos más de una vez.

$\sum_{i=1}^{n} T_{INTERRUPCION_i}$ es el tiempo total de interrupciones.

T_{DATOn} es el tiempo del dato transmitido una sola vez.

Si analizamos las ecuaciones (1) y (2) podemos concluir que:

- En la ecuación (1) el número de tramas no retransmitidas será siempre mucho menor en el peor escenario, debido a que las interrupciones ocurren cuando el usuario ha visto la mayor cantidad del video.
- En la ecuación (2) el número de tramas no retransmitidas será siempre mucho mayor en el mejor escenario, debido a que las interrupciones ocurren cuando el usuario se encuentra en los inicios del video.

Con la finalidad de evaluar el modelo matemático es necesario ponderar el peso que tiene el retransmitir un flujo en base al número de interrupciones dadas. Se plantean las siguientes consideraciones:

- Capacidad de canal constante.
- No existen retransmisiones por causas de pérdida de datos en el canal.

Ahora determinaremos la ecuación que permita calcular la cantidad de bits transmitidos en cada T_{DATOi}:

$$Número\ de\ bits\ transmitidos = T_{DATO_i} \times Capacidad\ de\ Canal$$

Si consideramos un canal ADSL con acceso mediante WiFi que tiene una capacidad del canal de 10 Mbps y si T_{DATO1} tuvo un tiempo de ejecución de 17,96 segundos tenemos que el número de bits que se transmitieron sería de:

$$Número\ de\ bits\ transmitidos = 17{,}96 \times 10 = 179{,}6\ Mbits = 22{,}45\ MB$$

Analicemos un ambiente en el que tenemos tres interrupciones, los tiempos de cada uno de los T_{FLUJO} generados por las interrupciones son: de 27, 13, 3 y 31 medidos en segundos y la capacidad del canal es 10 Mbps. Si aplicamos el método descrito para evaluar el mejor o peor escenario, tenemos los resultados que se indican en las tablas 2.5-2.6.

Tabla 2.5: Flujos ubicados y ordenados para escenario en (s)

	Tflujo3	Tflujo2	Tflujo1	Tflujo4	Número retransmisiones flujos
Tdato1 (s)	3	3	3	3	3
Tdato2 (s)		10	10	10	2
Tdato3 (s)			14	14	1
Tdato4 (s)				4	0
Tflujo (s)	3	13	27	31	
Cantidad de datos (MB)	3,75	16,25	33,75	38,75	

Tabla 2.6: Flujos escenario en (MB)

	Tflujo3	Tflujo2	Tflujo1	Tflujo4	Número retransmisiones flujo
Tdato1 (MB)	3,75	3,75	3,75	3,75	3
Tdato2 (MB)		15,625	15,625	15,625	2
Tdato3 (MB)			17,5	17,5	1
Tdato4 (MB)				5	0
Tflujo (MB)	3,75	16,25	33,75	38,75	
Capacidad canal (Mbps)	10	10	10	10	

Tabla 2.7: Total de bytes transmitidos y retransmitidos por flujo

Flujo	Tflujo (s)	Tflujo (MB)	Datos retransmitidos (MB)	Número retransmisiones	Total datos retransmitidos (MB)
1	27	33,75	20	1	20
2	13	16,25	7,5	2	15
3	3	3,75	3,75	3	11,25

Ahora calculemos el total de bytes trasmitidos en cada flujo e igual lo hacemos para los retransmitidos, los resultados se exponen en la tabla 2.7.

Ahora calculemos los totales del flujo completo de este escenario, descritos en la tabla 2.8.

De los resultados observamos que se ha retransmitido un total de datos de 53,75 MB y el total de datos de video es 38,75 MB, dejando un incremento de video transmitido de 138,71%.

Se presentan la Figura 2.23 que permiten visualizar el número de retransmisiones que ha tenido el flujo y asociado el total de datos retransmitidos en (MB).

La Figura 2.24 permite visualizar el total de datos del video versus el total de datos retransmitidos.

Tabla 2.8: Total de bytes transmitidos y retransmitidos por video

Total tiempo video (s)	Total datos video (MB)	Total datos retransmisiones (MB)	Incremento (%)
31	38,75	53,75	138,71

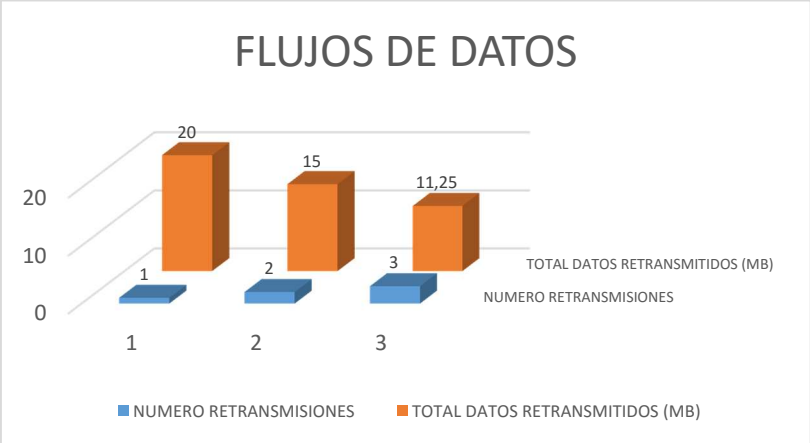


Figura 2.23: Datos transmitidos vs retransmitidos por flujo

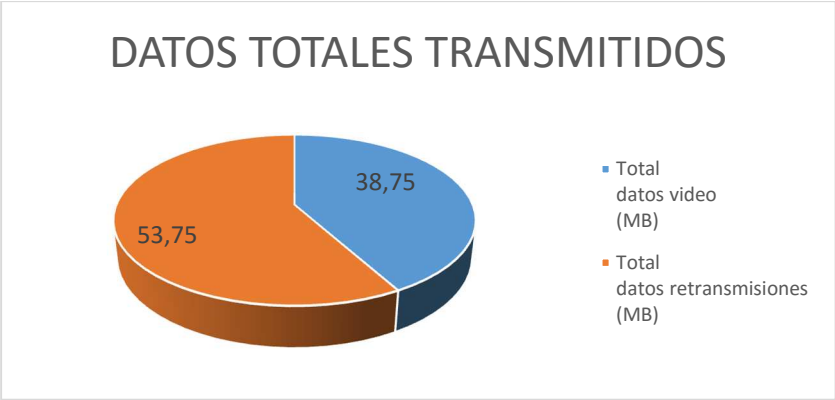


Figura 2.24: Datos totales transmitidos vs retransmitidos

Tabla 2.9: Flujos ubicados y ordenados para mejor escenario en (s)

	Tflujo2 (s)	Tflujo1 (s)	Tflujo3 (s)	Tflujo4 (s)	Número retransmisiones flujos
Tdato1 (s)	13,96	13,96	13,96	13,96	3
Tdato2 (s)		4	4	4	2
Tdato3 (s)			11,3	11,3	1
Tdato4 (s)				1383,74	0
Tflujo (s)	13,96	17,96	29,26	1413	
Cantidad de datos (MB)	17,45	22,45	36,575	1766,25	

Tabla 2.10: Flujos mejor escenario en (MB)

	Tflujo2 (s)	Tflujo1 (s)	Tflujo3 (s)	Tflujo4 (s)	Número retransmisiones flujo
Tdato1 (MB)	17,45	17,45	17,45	17,45	3
Tdato2 (MB)		5	5	5	2
Tdato3 (MB)			14,125	14,125	1
Tdato4 (MB)				1729,675	0
Tflujo (MB)	17,45	22,45	36,575	1766,25	
Capacidad canal (Mbps)	10	10	10	10	

Ahora realizamos el mismo análisis para un caso experimental del mejor escenario cuyos datos son: 17.96, 13.96 y 29.26 medidos en segundos y el tiempo completo del video es 1413 segundos.

Si aplicamos el método descrito para evaluar el mejor escenario, tenemos los resultados que se indican en las tablas 2.9-2.10.

Ahora calculemos el total de bytes trasmitidos en cada flujo e igual lo hacemos para los retransmitidos, los resultados se exponen en la tabla 2.11.

Ahora calculemos los totales del flujo completo de este escenario, descritos en la tabla 2.12.

De los resultados observamos que se ha retransmitido un total de datos de 76,48 MB y el total de datos de video es 1766,25 MB, dejando un incremento de video transmitido de 4,33%.

Se presenta la Figura 2.25 que permiten visualizar el número de retransmisiones que ha tenido el flujo y asociado el total de datos retransmitidos en (MB).

La Figura 2.26 permite visualizar el total de datos del video versus el total de datos retransmitidos.

Tabla 2.11: Total de bytes transmitidos y retransmitidos por flujo

Flujo	Tflujo (s)	Tflujo (MB)	Datos retransmitidos (MB)	Número retransmisiones	Total datos retransmitidos (MB)
1	29,26	36,575	39,9	1	39,9
2	17,96	22,45	34,9	2	69,8
3	13,96	17,45	17,45	3	52,35

Tabla 2.12: Total de bytes transmitidos y retransmitidos por video

Total tiempo video (s)	Total datos video (MB)	Total datos retransmisiones (MB)	Incremento (%)
1413	1766,25	76,48	4,33

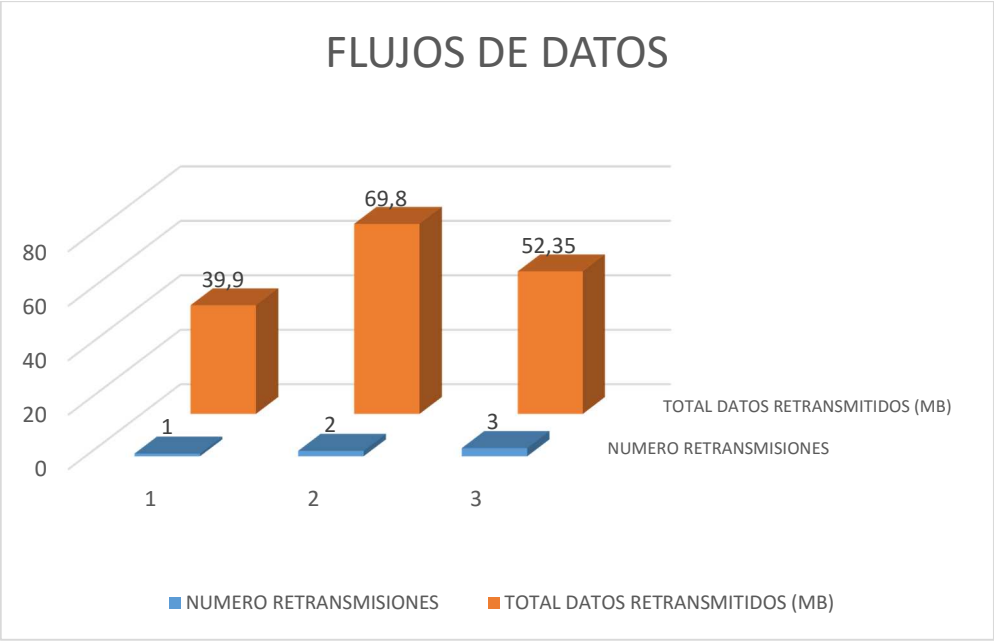


Figura 2.25: Datos transmitidos vs retransmitidos por flujo

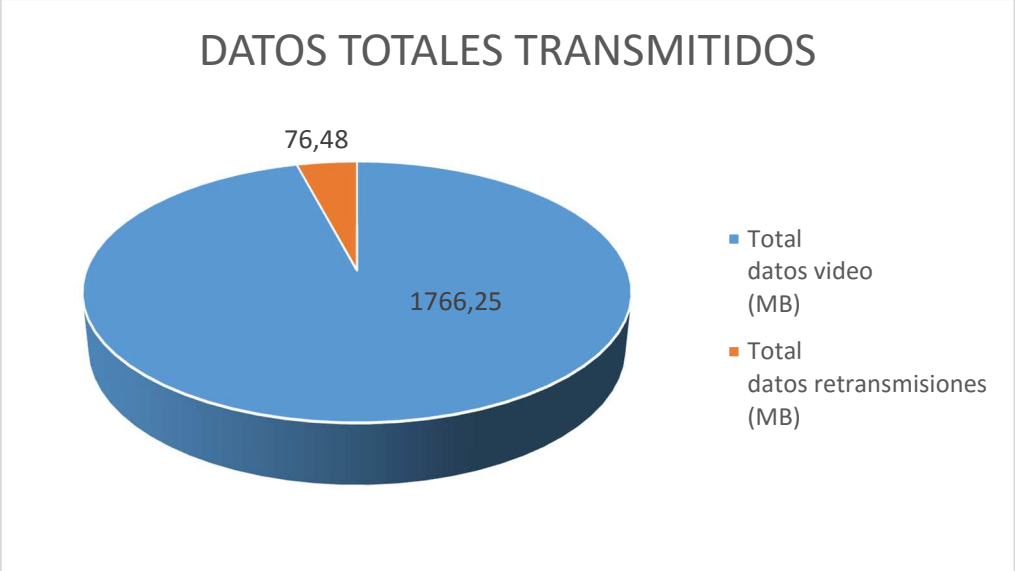


Figura 2.26: Datos totales transmitidos vs retransmitidos

Ahora realizamos el mismo análisis para un caso experimental del peor escenario cuyos datos son: 114.44, 123.48 y 157.61 medidos en segundos y el tiempo completo del video es 1413 segundos, tenemos los resultados que se indican en las tablas 2.13-2.14.

Tabla 2.13: Flujos ubicados y ordenados para peor escenario en (s)

	Tflujo1 (s)	Tflujo2 (s)	Tflujo3 (s)	Tflujo4 (s)	Número retransmisiones flujos
Tdato1 (s)	114,44	114,44	114,44	114,44	3
Tdato2 (s)		9,04	9,04	9,04	2
Tdato3 (s)			34,13	34,13	1
Tdato4 (s)				1255,39	0
Tflujo (s)	114,44	123,48	157,61	1413	
Cantidad de datos (MB)	143,05	154,35	197,0125	1766,25	

Tabla 2.14: Flujos peor escenario en (MB)

	Tflujo1 (s)	Tflujo2 (s)	Tflujo3 (s)	Tflujo4 (s)	Número retransmisiones flujo
Tdato1 (MB)	143,05	143,05	143,05	143,05	3
Tdato2 (MB)		11,3	11,3	11,3	2
Tdato3 (MB)			42,6625	42,6625	1
Tdato4 (MB)				1569,2375	0
Tflujo (MB)	143,05	154,35	197,0125	1766,25	
Capacidad canal (Mbps)	10	10	10	10	

Tabla 2.15: Total de bytes transmitidos y retransmitidos por flujo

Flujo	Tflujo (s)	Tflujo (MB)	Datos retransmitidos (MB)	Número retransmisiones	Total datos retransmitidos (MB)
1	157,61	197,0125	297,4	1	297,4
2	123,48	154,35	286,1	2	572,2
3	114,44	143,05	143,05	3	429,15

Ahora calculemos el total de bytes trasmitidos en cada flujo e igual lo hacemos para los retransmitidos, los resultados se exponen en la tabla 2.15.

Ahora calculemos los totales del flujo completo de este escenario, descritos en la tabla 2.16.

De los resultados observamos que se ha retransmitido un total de datos de 494,41 MB y el total de datos de video es 1766,25 MB, dejando un incremento de video transmitido de 27,99%.

Se presenta la Figura 2.27 que permiten visualizar el número de retransmisiones que ha tenido el flujo y asociado el total de datos retransmitidos en (MB).

La Figura 2.28 permite visualizar el total de datos del video versus el total de datos retransmitidos.

Tabla 2.16: Total de bytes transmitidos y retransmitidos por video

Total tiempo video (s)	Total datos video (MB)	Total datos retransmisiones (MB)	Incremento (%)
1413	1766,25	494,41	27,99

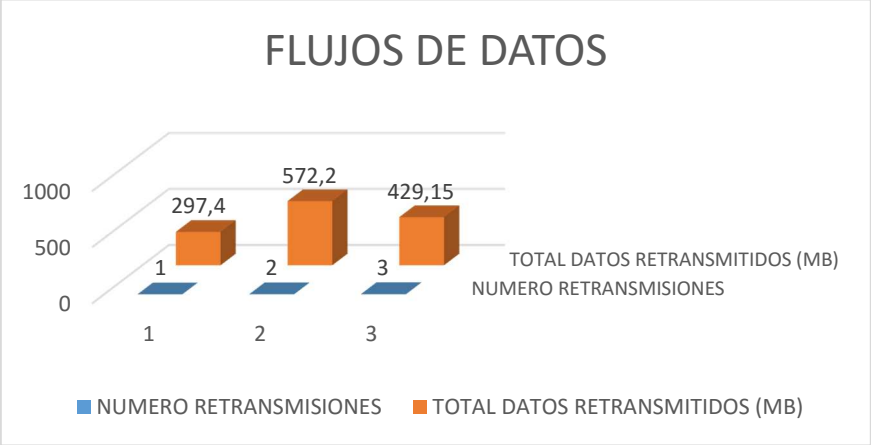


Figura 2.27: Datos transmitidos vs retransmitidos por flujo

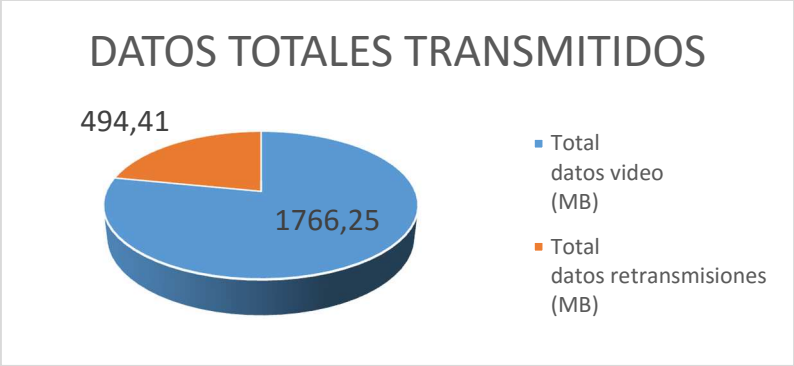


Figura 2.28: Datos totales transmitidos vs retransmitidos

Tabla 2.17: Resultados mejor escenario

MEJOR ESCENARIO										
PRUEBA	Tinicio	Tflujo1	Tflujo2	Tflujo3	Tflujo4	Tint1	Tint2	Tint3	Tflujovideo	Tejecucion
1	2,8	30,61	50,2	34,35	1297,84	21,11	9,54	8,98	1413	1514,03
2	2,83	30,75	30,61	32,79	1318,85	8	9,35	9,8	1413	1484,26
3	2,66	18,8	18,13	10,04	1366,03	11,86	11,55	14,83	1413	1480,68
4	3,06	29,63	24,46	30,24	1328,67	12,28	13,16	10,9	1413	1491,82
5	3,93	16,2	22,18	11,8	1362,82	12,21	10,9	16,06	1413	1490,07
6	2,2	17,96	13,96	29,26	1351,82	11,83	13,41	9,06	1413	1485,36

Tabla 2.18: Resultados peor escenario

PEOR ESCENARIO										
PRUEBA	Tinicio	Tflujo1	Tflujo2	Tflujo3	Tflujo4	Tint1	Tint2	Tint3	Tflujovideo	Tejecucion
1	2,28	144,01	147,45	217,51	904,03	12,29	11,63	11,78	1413	1675,33
2	2,43	189,5	197,79	223,81	801,9	10,96	8,3	13,84	1413	1679,63
3	2,7	182,76	152,16	234,81	843,27	12,69	8,81	15,38	1413	1695,49
4	5,05	114,44	123,48	157,61	1017,47	22,9	12,14	16,38	1413	1642,23
5	3,18	81,26	115,08	51,98	1164,68	12,11	13,31	9,44	1413	1575,66
6	4,7	73,83	130,88	83,13	1125,16	14,99	12,23	18,54	1413	1608,44

A continuación se analiza el tiempo de ejecución de los escenarios ideal, mejor y peor. Se indica los resultados obtenidos de seis experimentos con un video de duración de 1413 segundos y aplicando tres interrupciones. Se utilizó como servidor de video streaming YouTube, una conexión ADSL de 10Mbps y un dispositivo móvil Samsung Galaxy S4. Los resultados se encuentran en las tablas 2.17-2.18.

Del análisis de los escenarios y sustentado en la Figura 2.29, se observa claramente cómo impacta el número de retransmisiones de flujos en el canal de comunicaciones, por la generación de tráfico innecesario en la red y con posibles congestiones del canal. Los tiempos de espera para acceder a recurso de la red, por este tráfico innecesario en el canal, causa malestar en el usuario. Por lo tanto la calidad de experiencia del usuario es mala.

De la gráfica podemos apreciar que para el escenario peor presenta drásticas variaciones, quiere decir que mientras más cercano me encuentro a la finalización del video, mayor penalidad tendré en retransmisiones de los flujos. Mientras que en el escenario mejor tenemos una variación mínima aceptable. Es un escenario recomendado para uso en dispositivos que no dispongan de grandes cantidades de almacenamiento temporal. Desde el punto de vista de la QoE, impacta significativamente de manera negativa al usuario por el hecho de tener que visualizar repetidas veces los mismos flujos de datos del video.

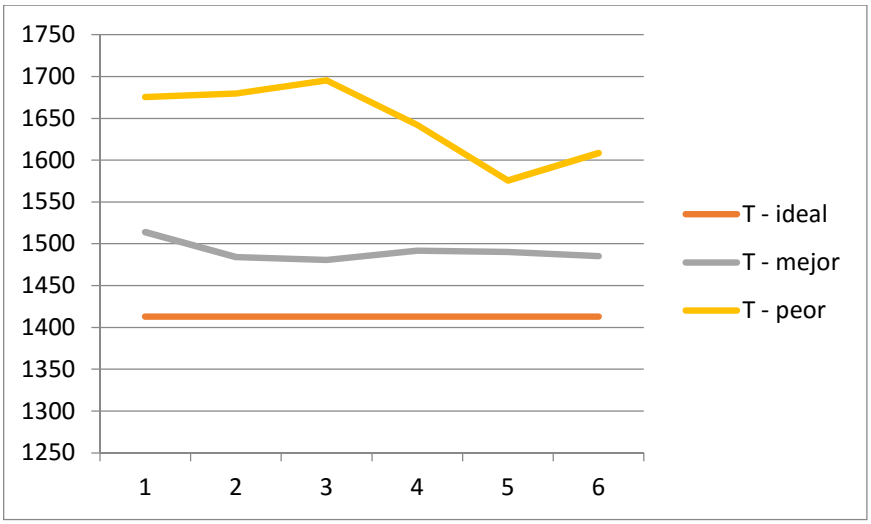


Figura 2.29: Tiempo de ejecución escenarios

Para hacer pruebas experimentales para verificar el modelo de tiempo diseñado, procedimos a hacer pruebas en el Pasillo del Pabellón C del Edificio de Electrónica y Telecomunicación (sede del Departamento de Ingeniería Telemática).

En primer lugar intentamos acceder a un vídeo almacenado con la aplicación de youtube de tres móviles diferentes (NEXUS 5, iPhone y Samsumg Galaxy 2). Para video almacenado se producía un efecto buffer que evitó que se pudieran hacer pruebas de conexión y reconexión porque el vídeo se almacenaba por completo en el móvil y ya no se podía medir las desconexiones.

Luego probamos con la página Web de la Radio Televisión Española (rtve.com) para descargar vídeo en directo. Pero el problema esta vez es que cuando el vídeo en directo se cortaba, ya no era posible volver a reconectar. Era necesario volver a abrir una sesión nueva.

Hemos de notar que a nosotros no nos interesa si el video se recupera o no, lo que nos interesa es observar los tiempos: el de comienzo (T_{INICIO}), en los que se transmiten datos (T_{DATO}) y en los que no hay tal transmisión ($T_{INTERRUPCION}$).

Por esta razón decidimos utilizar una aplicación propia que si nos permitía medir el efecto de estas reconexiones, en nuestro caso lo hicimos utilizando la plataforma

WebRTC-ULPGC; se realizaron 6 experimentos aplicando 3 interrupciones con los siguientes resultados reales (Tabla 2.19).

Ahora se procede a calcular el tiempo de ejecución utilizando el modelo matemático y los resultados son los siguientes (Tabla 2.20):

Se puede observar en la Figura 2.30, la afectación clara de las interrupciones en el tiempo de ejecución del video streaming, generando hasta el 90% adicional del tiempo de duración del video lo que afecta a la QoE del cliente.

Tabla 2.19: Resultados reales utilizando la plataforma WebRTC-ULPGC

PRUEBA	T1	T2	T3	T4	T5	T6	T7	T8	Duración Total	Duración segundos
1	00:4,54	00:47,45	00:41,78	00:17,51	00:32,11	00:44,01	00:38,98	00:15,04	04:01,38	241,38
2	00:4,43	00:47,79	00:33,84	00:43,81	00:31,8	00:49,5	00:32,8	00:15,08	04:19,02	259,02
3	00:4,76	00:52,16	00:35,38	00:34,81	00:35,86	00:42,76	00:34,83	00:15,02	04:15,56	255,56
4	00:4,26	00:23,48	00:36,38	00:57,61	00:32,28	00:14,44	00:30,9	00:15,03	03:34,35	214,35
5	00:5,09	00:15,08	00:29,44	00:51,98	00:22,21	00:41,26	00:26,06	00:15,01	03:26,10	206,10
6	00:5,22	00:34,54	00:31,23	00:37,67	00:31,83	00:37,34	00:39,06	00:15,04	03:51,89	231,89

Tabla 2.20: Aplicación del modelo matemático a los resultados obtenidos

PRUEBA	TINICIO	TDATO1	TINT1	TDATO2	TINT2	TDATO3	TINT3	TDATO4	Sumatorio TDATO	Sumatorio TINT	TEJECUCION	DIFERENCIA
1	4,54	47,45	41,78	17,51	32,11	44,01	38,98	15,04	108,97	112,87	241,42	0,04
2	4,43	47,79	33,84	43,81	31,80	49,50	32,80	15,08	141,10	98,44	259,05	0,03
3	4,76	52,16	35,38	34,81	35,86	42,76	34,83	15,02	129,73	106,07	255,58	0,02
4	4,26	23,48	36,38	57,61	32,28	14,44	30,90	15,03	95,53	99,56	214,38	0,03
5	5,09	15,08	29,44	51,98	22,21	41,26	26,06	15,01	108,32	77,71	206,13	0,03
6	5,22	34,54	31,23	37,67	31,83	37,34	39,06	15,04	109,55	102,12	231,93	0,04

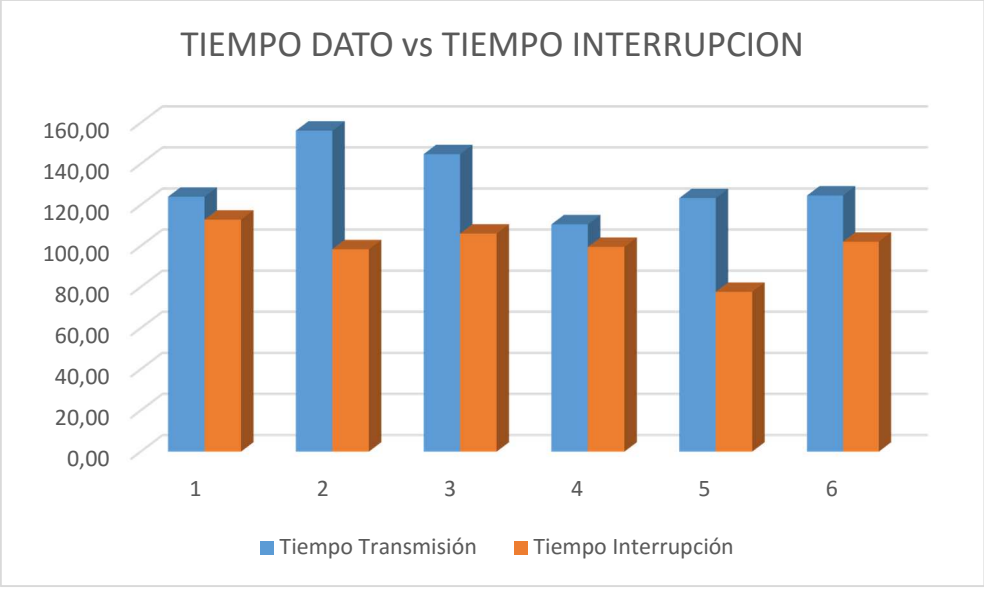


Figura 2.30: Tiempo Dato y Tiempo Interrupción

3. Solución base para control de interrupciones

En este capítulo se presenta un modelo basado en patrones software de diseño de software que implementa un mecanismo que mitiga las interrupciones de la sesión de servicio video streaming establecida ante una disrupción en el canal inalámbrico en ambientes Android. Este mecanismo utiliza agentes inteligentes basados en JADE e implementados por el add-on JADE-Android, para controlar una interrupción de una sesión video streaming de video establecida en un dispositivo móvil Android y la continuidad del video streaming, una vez que se restablece la comunicación inalámbrica.

3.1. Análisis del diseño basado en patrones software

En el capítulo 2 se planteó el modelo del servicio video streaming mediante patrones software de diseño de software, concretamente patrón MVC. Posterior se realizó el modelamiento matemático que permitió demostrar que existe un impacto negativo en el servicio de video streaming, debido a la disrupción del canal de comunicación inalámbrico que provoca la interrupción del servicio. Este impacto negativo, en el lado de la red genera tráfico y congestión por la cantidad de retransmisiones de datos solicitadas, con la posibilidad de saturar la capacidad del canal. Con respecto al usuario, genera una negativa calidad de experiencia por las continuas interrupciones que ocasionan reinicios del servicio permanentemente, demoras en el restablecimiento de la sesión y visualización de repetitiva de flujos de datos; que pueden determinar en el usuario el abandono definitivo de la sesión video streaming.

Nuestro interés es establecer un modelo basado en patrones software de diseño que permita controlar las interrupciones del servicio video streaming ocasionadas por quiebres del canal de comunicación inalámbrico. Permitiendo así mitigar o minimizar el tiempo de restablecimiento de la sesión video streaming, la retransmisión de flujos de datos y en consecuencia la no visualización repetitiva de estos flujos.

Con esto se pretende validar que nuestro modelo reduce considerablemente el tiempo de establecimiento de la sesión, retransmisiones y en consecuencia el tiempo total de ejecución del servicio. Esto mejoraría notablemente la QoE del usuario debido a que el usuario va a tener menores tiempos de restablecimiento de la sesión y aportando continuidad al flujo de datos del video.

Para conseguir esto, nuestro modelo incorpora funcionalidades que permiten controlar las interrupciones de la sesión por quiebres del canal y reconectar automáticamente, garantizando la continuidad del servicio video streaming. Esto permite además la continuidad de los flujos de datos del video mejorando la QoE al usuario.

Para mantener flexibilidad de cambio, orden y reutilización en el modelo propuesto, se utilizó el patrón de diseño MVC. Se utiliza el modelo de caso de usos para

describir la funcionalidad del modelo base. En la Figura 3.1 y Tabla 3.1-3.6 se muestra la relación entre los actores y los casos de en este modelo.

El primer caso de uso describe el inicio de petición de video y su despliegue en pantalla, explicado en la Tabla 3.1.

El segundo caso de uso describe la importancia de monitorear el umbral de almacenamiento temporal, explicado en la Tabla 3.2.

El tercer caso de uso describe el proceso a seguir para confirmar que se trata de una interrupción, se evalúa el estado del canal de comunicación, explicado en la Tabla 3.3.

El cuarto caso de uso describe como dado un evento del canal de comunicación, debe gestionarse la continuidad del servicio de video streaming, explicado en la Tabla 3.4.

El quinto caso de uso describe las actividades a cumplir cuando ocurre una interrupción y como se controla la misma, explicado en Tabla 3.5.

El sexto caso de uso describe las actividades a cumplir cuando el canal de comunicación se ha restablecido, explicado en la Tabla 3.6.

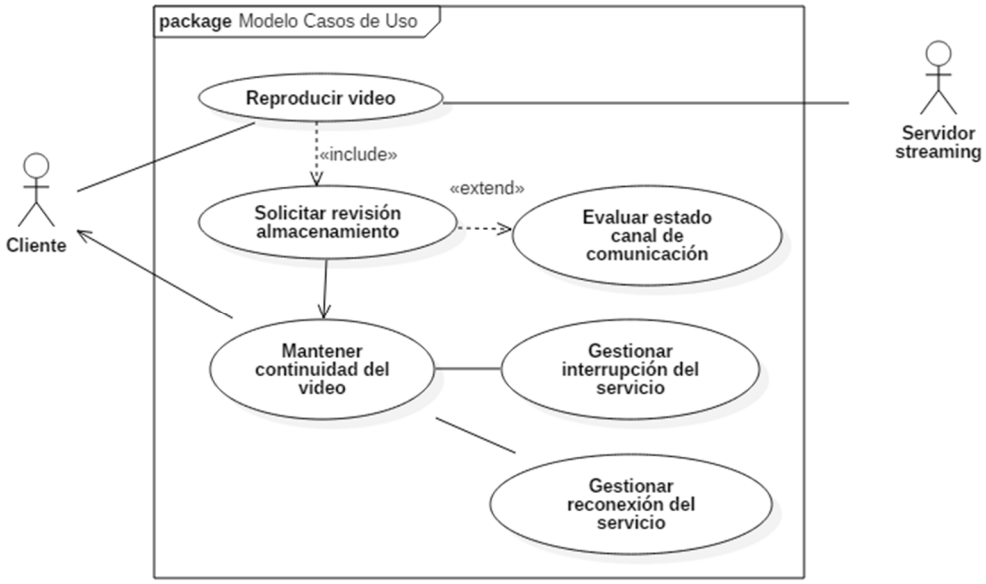


Figura 3.1: Casos de uso modelo base

Tabla 3.1: Casos de uso modelo base reproducir video

Caso de Uso: reproducir video
Actores: cliente
Resumen: describe inicio petición de video y su despliegue.
Precondiciones:
Descripción: 1. El cliente solicita visualización del video. 2. Ingresa datos IP del servidor, puerto y video. 3. Los datos son validados por el *Proxy*. 4. Si no existen problemas el *Proxy* solicita el servicio al servidor video streaming. 5. Se despliega una ventana en la que inicia el video streaming. 6. El servidor inicia la entrega de flujos al almacenamiento temporal.
Pos condiciones: solicita la revisión constante del almacenamiento.
Observaciones:

Tabla 3.2: Casos de uso modelo base solicitar revisión almacenamiento

Caso de Uso: solicitar revisión almacenamiento
Actores:
Resumen: explica la importancia de monitorear umbral de almacenamiento temporal
Precondiciones: el *Proxy* solicita la revisión del almacenamiento
Descripción: 1. Se pide constantemente información del umbral del almacenamiento. 2. Se compara con el máximo nivel establecido. 3. Se genera una alerta en caso de superación del umbral definido. 4. Se comunica este problema al *Proxy*
Pos condiciones: solicita la evaluación del canal de comunicación
Observaciones:

Tabla 3.3: Casos de uso modelo base evaluar estado de canal

Caso de Uso: evaluar estado canal de comunicación
Actores:
Resumen: para confirmar que se trata de una interrupción, se evalúa el estado del canal de comunicación.
Precondiciones: *Proxy* solicita evaluación del canal de comunicación
Descripción: 1. Se evalúa el canal de comunicación. 2. Se determina el estado del canal de comunicación 3. Este estado es comunicado al *Proxy*.
Pos condiciones: se solicita mantener la continuidad del servicio
Observaciones:

Tabla 3.4: Casos de uso modelo base mantener continuidad del video

Caso de Uso: mantener continuidad del video
Actores:
Resumen: dado un evento del canal de comunicación debe gestionarse la continuidad del servicio video streaming.
Precondiciones: el proxy solicita ejecución de proceso de acuerdo al estado del canal de comunicación.
Descripción: Alerta interrupción : 1. En caso de ser una alerta de interrupción se solicita la gestionar interrupción del servicio. Alerta conexión: 2. Cuando el canal se ha restablecido se solicita gestionar reconexión.
Pos condiciones:
Observaciones:

Tabla 3.5: Casos de uso modelo base gestionar interrupción del servicio

Caso de Uso: gestionar interrupción del servicio
Actores:
Resumen: explica las actividades a cumplir cuando ocurre una interrupción y como se controla la misma.
Precondiciones:
Descripción: 1. Se solicita el almacenamiento de los flujos en el buffer. 2. Se despliega en la interfaz gráfica del usuario el mensaje “Reconectando” por el problema dado y que se encuentra en estado de espera para reconectarse automáticamente 3. Se pide el servicio de evaluar continuamente el canal de comunicación.
Pos condiciones:
Observaciones:

Tabla 3.6: Casos de uso modelo base gestionar reconexión

Caso de Uso: gestionar reconexión
Actores:
Resumen: explica las actividades a cumplir cuando el canal se ha restablecido.
Precondiciones:
Descripción: 1. Se solicita que los flujos almacenados en el buffer sean transmitidos al dispositivo cliente. 2. El cliente automáticamente continúa visualizando el video en el punto en el que se encontraba cuando ocurrió la interrupción.
Pos condiciones:
Observaciones:

3.1.1. Justificación de los patrones software a utilizar

El propósito de este diseño basado en patrones software probados es generar un modelo genérico que pueda ser implementado en cualquier escenario de experimentación. Esto se consigue gracias a la utilización del patrón MVC que permite un desarrollo modular, escalable, flexible y de fácil mantenimiento. Se fortalece este patrón con la implementación de algunos patrones software adicional que permiten entregar más funcionalidades. Los patrones software adicionados son: *Composite*, *Observer*, *Strategy* y *Proxy*. En la capa vista un patrón *Composite* para organizar estructuradamente su contexto y componentes y facilidad a cambios. En la capa modelo un patrón *Observer* que acompañe para determinar oportunamente los cambios de estado del almacenamiento temporal. En la capa controlador un patrón *Strategy* que permita determinar la acción a seguir notificados por el patrón *Observer*, y un patrón *Proxy* que es el orquestador e intermediario del mecanismo control de interrupción entre el servidor y el cliente.

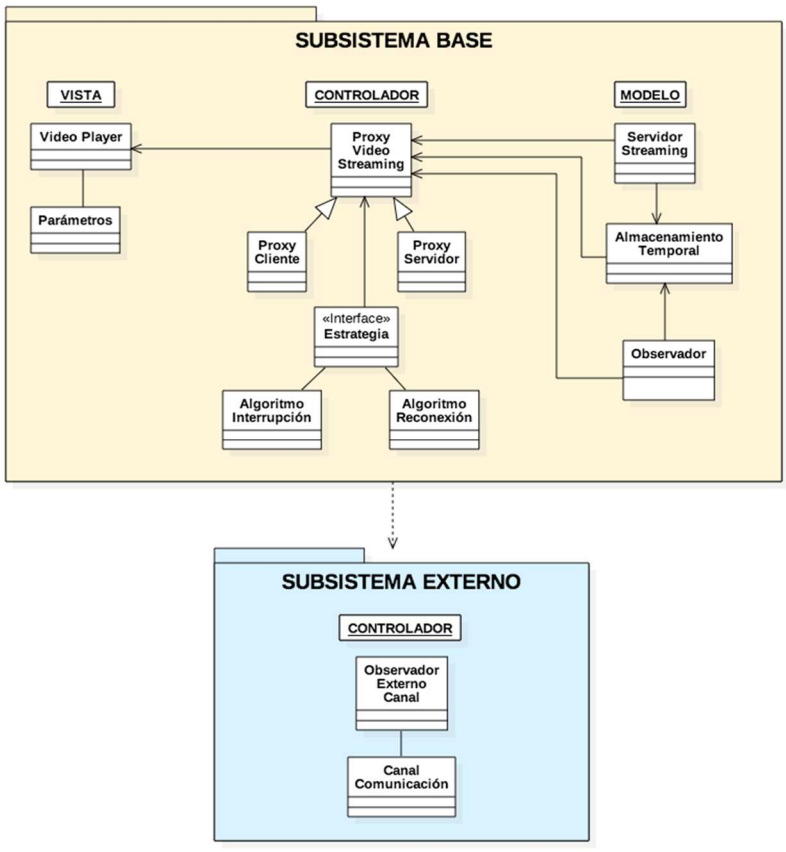


Figura 3.2: Modelo base para control de interrupciones

La Figura 3.2 indica cómo se encuentran ubicadas las funcionalidades del modelo base y el incremental que tiene para poder tener un mecanismo para control de interrupciones efectivo.

De acuerdo a los casos de uso tenemos que los patrones software se han implementado de la siguiente manera:

- *Reproducir video:* se ha realizado mediante el patrón *Composite*, permitiendo organizar estructuradamente su contexto y componentes y brindar facilidad a cambios.
- *Solicitar revisión almacenamiento:* se ha implementado mediante el patrón *Observer* y se encarga de monitorear el umbral del almacenamiento temporal, notificando una alerta al *Proxy* cuando supera el límite establecido.
- *Evaluar estado canal de comunicación:* entregada la alerta del *Observer* del almacenamiento temporal, se requiere validar si ésta es por quiebre del canal de comunicación. Esto se lo implementa ubicando un patrón *Observer* a nivel de la capa controlador que permita monitorear al canal y verificar si existe una interrupción, solicitado por *Proxy*.

Mantener continuidad del video: este caso de uso engloba a dos casos de uso *Gestionar interrupción del servicio* y *Gestionar reconexión del servicio*. Entregada la alerta de canal desconectado, *Proxy* solicita al patrón *Strategy*, implementado en caso de uso *Mantener continuidad del video* para que determine la acción a seguir de acuerdo a la alerta. Si la alerta es canal desconectado se procede con la acción descrita en el caso de uso *Gestionar interrupción del servicio* y se ejecuta la acción interrupción. Ésta acción solicita al servidor enviar los flujos de datos al almacenamiento temporal mientras se mantenga la alerta y despliegue en la vista el mensaje *Reconectando*. Superado el problema de interrupción, *Observer* del canal emite alerta canal conectado. *Proxy* solicita a *Strategy* determine la acción correspondiente y proceda con la acción descrita en el caso de uso *Gestionar reconexión del servicio* y ejecuta la acción reconexión. Ésta acción indica que los flujos almacenados en el buffer sean transmitidos al cliente.

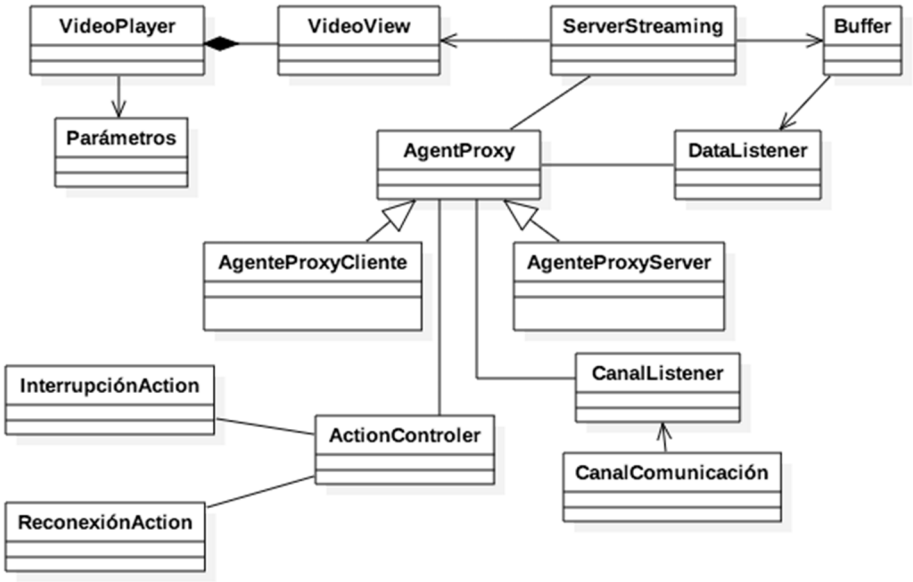


Figura 3.3: Diagrama de clases modelo base

3.1.2. Diseño arquitectónico y de despliegue

Basado en el modelo creado mediante patrón MVC se establece el diagrama de clases que se muestra en la Figura 3.3. Cada una de estas clases se encuentra asociada a una de las capas de MVC con sus respectivos patrones software así tenemos:

- Capa modelo y patrón *Observer*: engloban las clases que determinan el almacenamiento temporal y monitoreo el estado del umbral del buffer. Estas son: *AgentProxy*, *ServerStreaming*, *DataListener* y *Buffer*.

- Capa vista y patrón *Composite*: contiene las clases que permiten interactuar y desplegar el video con el servidor video streaming. Estas son: *Video Player*, *Parámetros* y *Video View*.

- Capa controlador y patrones software *Proxy* y *Strategy*: contiene las clases para controlar las interrupciones y tomar acciones. Estas son: *AgenteProxyCliente*, *AgenteProxyServidor*, *ActionControler*, *InterrupciónAction* y *ReconexiónAction*.

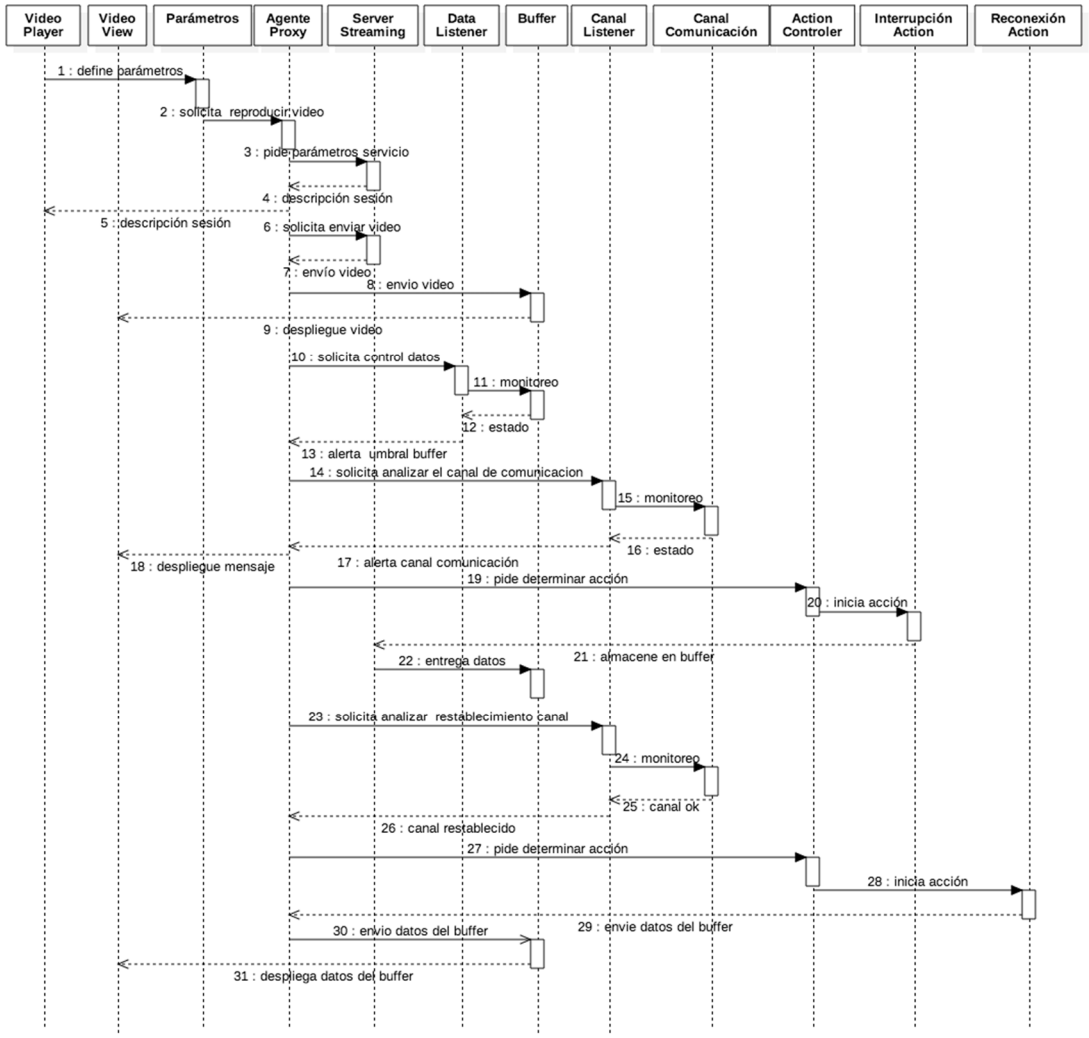


Figura 3.4: Diagrama de secuencia modelo base

El observador externo y patrón *Observer*: contiene las clases que evalúan el estado del canal de comunicación. Estas son: *CanalListener* y *CanalComunicación*.

La Figura 3.4 describe el diagrama de secuencia del modelo base diseñada, para la interacción entre el cliente y el servidor al solicitar la visualización de un video.

El *Video Player* es quien inicia la aplicación, definiendo los parámetros mediante la clase Parámetros. Posterior solicita reproducir video y eleva la petición a *Proxy* quien valida y pide parámetros de servicio al *ServidorStreaming*. Éste entrega descripción de sesión al *Proxy* para que los envíe a *Video Player*. Una vez establecida la sesión, *Proxy* solicita a *ServerStreaming* envíe video. *Buffer* recibe el flujo de datos e inmediato entrega a *VideoView* para su despliegue. Por su parte *Proxy* solicita control de datos a

DataListener sobre estado de umbral del *Buffer*. *Buffer* envía estado a *DataListener* y éste evalúa, determinando si existe alerta. Al darse alerta informa a *Proxy* para que confirme mediante monitoreo al canal de comunicación con *CanalListener*. Éste monitorea el canal de comunicación y entrega estado conectado o desconectado a *Proxy*. Si el estado entregado es canal desconectado inmediato *Proxy* solicita acción a *ActionControler*, quien determina la acción a realizarse. Con el estado de canal desconectado *ActionControler* invoca a *InterrupciónAction* para que ejecute acciones. La acción a ejecutarse es indicar al servidor que envíe el flujo de datos al *Buffer* mientras se mantenga es estado desconectado. *Proxy* solicita seguir monitoreando al canal e informe si existe cambio de estado. Al presentarse cambio de estado a canal conectado, vuelve a solicitar acción a *ActionControler*, quien invoca a *ReconexiónAction* para que ejecute acciones. La acción es indicar que *Buffer* envíe los flujos almacenados para que *VideoView* despliegue el video.

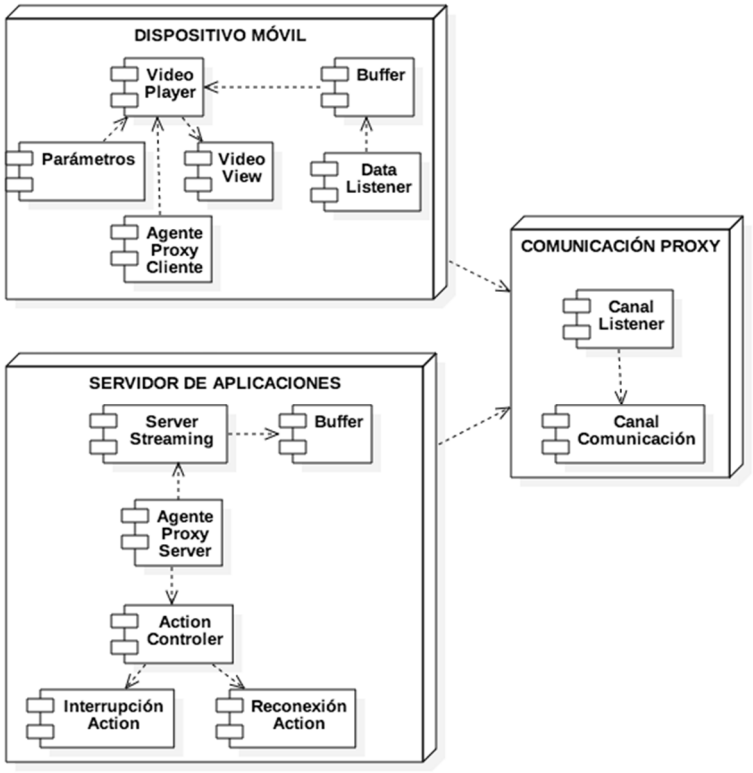


Figura 3.5: Diagrama de despliegue modelo base

La Figura 3.5 describe el diagrama de despliegue del servicio de video streaming con control de interrupciones. Contiene dos usuarios: cliente y servidor. El cliente corresponde a cualquier dispositivo móvil y contiene las clases: *VideoPlayer*, *VideoView*, *Parámetros*, *AgenteProxyCliente*, *DataListener* y *Buffer*. El servidor es un servidor de aplicaciones que contiene las clases: *ServerStreaming*, *AgenteProxyServidor*, *ActionControler*, *InterrupciónAction*, *ReconexiónAction* y *Buffer*. Se debe indicar que la clase *AgentProxy* de la capa controlador se encuentra distribuida entre el cliente y el servidor con el propósito de atender sus peticiones y respuestas. Esta comunicación entre estos agentes de *AgentProxy* establece un canal de comunicación que lo monitorea *CanalListener*.

Utilizando el diagrama de despliegue, explicado anteriormente, *VideoPlayer* instancia mediante mensaje interno a *Parámetros* para obtener información para solicitar video. Con esta información Video Player envía una petición mediante protocolo RTSP pidiendo parámetros de sesión al *Proxy*. Éste solicita a *Servidor* estos parámetros. *Servidor* mediante protocolo SDP envía la descripción de la sesión al cliente. Es preciso indicar que la comunicación entra el cliente y el servidor ahora tiene un intermediario que es *Proxy*. Establecida la sesión, *Proxy* solicita mediante mensaje interno al servidor enviar el video de la fuente indicada en *URL Streaming*.

El servidor entrega el flujo de datos del video a *Proxy* mediante mensaje interno. *Proxy* envía el flujo de datos mediante protocolo RTP al *Buffer* del cliente. El *Buffer* mediante mensaje interno entrega al *VideoView* el flujo de datos para su visualización. La comunicación entre agentes proxy del cliente y del servidor lo realiza con mensajes FIPA-ACL.

3.2. Modelo matemático de rendimiento

Partiendo de la ecuación obtenida en el modelo matemático para el servicio de video streaming:

$$T_{EJECUCION} = (n+1) \times T_{INICIO_{FLUJO}} + \sum_{i=1}^{n} \left((i+1) \times T_{DATO_{n+1-i}} \right) + \sum_{i=1}^{n} T_{INTERRUPCION_i} + T_{DATOn} \quad (1)$$

Donde:

$(n+1) \times T_{INICIO_{FLUJO}}$ es el tiempo total de inicio de los flujos transmitidos por n interrupciones.

$\sum_{i=1}^{n} \left((i+1) \times T_{DATO_{n+1-i}} \right)$ es el tiempo total de T_{DATO} transmitidos más de una vez.

$\sum_{i=1}^{n} T_{INTERRUPCION_i}$ es el tiempo total de interrupciones.

T_{DATOn} es el tiempo del dato transmitido una sola vez.

El modelo base establece su funcionalidad mediante el uso de patrones software de diseño, que implementan dos patrones software observador; uno para monitorear el umbral de almacenamiento temporal del dispositivo móvil y otro para validar el estado del canal de comunicación. Con esta implementación se puede controlar la interrupción. Si se presenta una interrupción el mecanismo pide almacenar los flujos emitidos por el servidor en un buffer; consiguiendo así minimizar los tiempos de inicio de sesión. Esto permite que ya no se tenga que empezar de nuevo la visualización del video, sino simplemente pedir la descarga de lo que este en el buffer, una vez que el canal de comunicación este operativo.

Con estas consideraciones, tenemos que solo tenemos un T_{INICIO} presentado por la primera vez que estableció la sesión, quedando la ecuación (1) así:

$$T_{EJECUCION} = T_{INICIO_{FLUJO}} + \sum_{i=1}^{n} \left((i+1) \times T_{DATO_{n+1-i}} \right) + \sum_{i=1}^{n} T_{INTERRUPCION_i} + T_{DATOn}$$

Considerando que el mecanismo pide almacenar los flujos no transmitidos debido a una interrupción, para su posterior envío; este proceso anula las retransmisiones de flujos. Esto permite plantear lo siguiente:

En el modelo inicial la transmisión de datos con más de una vez está determinado por:

$$T_{RETRANSMISION} = \sum_{i=1}^{n} \left((i+1) \times T_{DATO_{n+1-i}} \right)$$

Quitando el sumatorio:

$$T_{RETRANSMISION} = (n+1) \times T_{DATO1} + n \times T_{DATO2} + (n-1) \times T_{DATO3} + \cdots + 2 \times T_{DATOn-1}$$

Como ya no existen retransmisiones los T_{DATO} solo se transmiten una vez, entonces $T_{RETRANSMISION}$ = 0 y solo queda el flujo del video. En la ecuación tenemos entonces:

$$T_{EJECUCION} = T_{INICIO_{FLUJO}} + \sum_{i=1}^{n} T_{DATO_i} + \sum_{i=1}^{n} T_{INTERRUPCION_i} + T_{DATOn}$$

Donde:

T_{INICIO} es el tiempo de inicio de solicitud del servicio de video streaming.

$\sum_{i=1}^{n}(T_{DATO_i})$ es el tiempo total de T_{DATO} transmitidos una sola vez.

$\sum_{i=1}^{n} T_{INTERRUPCION_i}$ es el tiempo total de interrupciones.

T_{DATO_n} es tiempo del último T_{DATO} transmitido.

Se realiza la evaluación del modelo matemático planteado con un video de duración de 983 segundos y ejecutado 3 interrupciones, se han realizado pruebas en dos tipos de escenarios: peor y mejor.

Los datos experimentales para el mejor escenario se describen en la tabla 3.7. Los datos experimentales para el peor escenario se describen en la tabla 3.8.

Se realizó la gráfica de los escenarios (Figura 3.6) y al compararla con la obtenida en el modelo planteado para el servicio video streaming tenemos las siguientes conclusiones:

- Mejora notablemente el tiempo de ejecución debido a que no tenemos tiempos acumulados por retransmisiones de flujos.
- En este modelo no influye si las interrupciones se dan al inicio o al final del video.
- Se demuestra que el tiempo de ejecución es menor con la utilización del modelo base para control de interrupciones, debido a que no penalizamos con tiempos de retransmisión de flujos.
- Es importante considerar que el modelo se sustenta en utilizar un almacenamiento temporal en el servidor de aplicaciones, para almacenar los flujos que no se pueden enviar al cliente por interrupción del canal de comunicación. Entonces ahora es necesario analizar el coste que me implicaría y si impacta en el modelo.

Tabla 3.7: Datos experimentales mejor escenario

MEJOR ESCENARIO										
PRUEBA	Tinicio	Tflujo1	Tflujo2	Tflujo3	Tflujo4	Tint1	Tint2	Tint3	Tflujovideo	Tejecucion
1	3,06	29,63	24,46	30,24	898,67	12,28	13,16	10,9	983	1022,4
2	2,66	18,8	18,13	10,04	936,03	11,86	11,55	14,83	983	1023,9
3	2,2	17,96	13,96	29,26	921,82	11,83	13,41	9,06	983	1019,5
4	3,93	16,2	22,18	11,8	932,82	12,21	10,9	16,06	983	1026,1
5	2,8	30,61	50,2	34,35	867,84	21,11	9,54	8,98	983	1025,43
6	2,83	30,75	30,61	32,79	888,85	8	9,35	9,8	983	1012,98

Tabla 3.8: Datos experimentales peor escenario

PEOR ESCENARIO										
PRUEBA	Tinicio	Tflujo1	Tflujo2	Tflujo3	Tflujo4	Tint1	Tint2	Tint3	Tflujovideo	Tejecucion
1	5,05	114,44	123,48	157,61	587,47	22,9	12,14	16,38	983	1039,47
2	3,18	81,26	115,08	51,98	734,68	12,11	13,31	9,44	983	1021,04
3	2,7	182,76	152,16	234,81	413,27	12,69	8,81	15,38	983	1022,58
4	2,43	189,5	197,79	223,81	371,9	10,96	8,3	13,84	983	1018,53
5	2,28	144,01	147,45	217,51	474,03	12,29	11,63	11,78	983	1020,98
6	4,7	73,83	130,88	83,13	695,16	14,99	12,23	18,54	983	1033,46

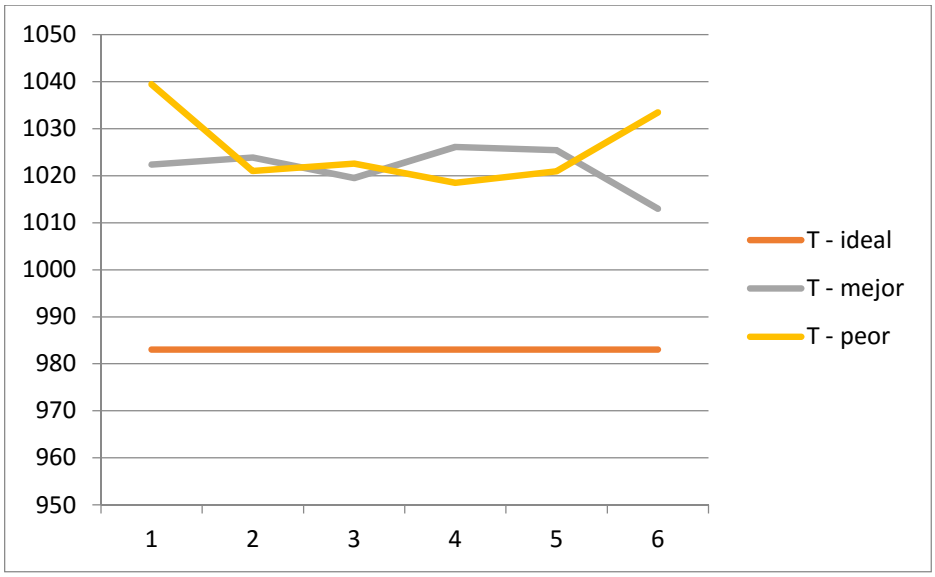


Figura 3.6: Tiempo de ejecución escenarios

La cantidad de memoria requerida en un escenario con un tiempo de interrupción de 22,9 segundos, que corresponde a la máxima interrupción presentada en el peor escenario experimental, y con una capacidad del canal ADSL de 10Mbps es:

$$Capacidad\ memoria = Capacidad\ canal \times Tiempor\ interrupción$$

$$Capacidad\ memoria = 10 \times 22{,}9$$

$$Capacidad\ memoria = 229\ Mbits$$

$$Capacidad\ memoria = \frac{229\ Mbits}{8} = 28{,}625\ MB$$

Entonces se concluye que la capacidad necesitada no afecta en coste al modelo. Además tenemos alternativas hoy en día como almacenamiento locales muy pero muy grandes, orden de los TB, y almacenamientos en la nube sin coste por el orden de los GB.

3.3. Análisis de la implantación experimental

La transmisión de video utiliza el mecanismo video streaming, el mismo permite la descarga parcial del video en pequeños fragmentos de menor tamaño en el cliente,

mejorando la usabilidad al enviar una secuencia de cuadros de video que presentan bajas demoras al iniciar la visualización y menores requisitos de almacenamiento.

Durante el proceso de transmisión del video, podrían existir rupturas o cortes especialmente en comunicación con redes inalámbricas; sin embargo, si se privilegia la usabilidad del sistema, es decir, que la descarga sea paulatina a medida en que divide en flujos el video a transmitir, sería deseable que al sufrir una interrupción el propio aplicativo identifique el restablecimiento de la comunicación y continúe el proceso de descarga sin que el usuario vuelva a reiniciar la sesión. Es en este escenario donde los sistemas basados en Agentes de Software como JADE, Figuran como alternativa para controlar y mejorar el proceso de transmisión de datos, incrementando de esta forma, la usabilidad del sistema.

Al requerir del funcionamiento de estos agentes de software en dispositivos inalámbricos se debe utilizar JADE-LEAP. El despliegue de este agente se lo realiza en modo partido, quiere decir que el contenedor en el que se ejecuta el agente se divide en una parte ubicada en el dispositivo móvil (Front-End) y otra parte ubicado en el servidor (Back-End), los cuales siempre están conectados.

JADE-LEAP posee agentes inteligentes que se encargan de cualquier tarea o comportamiento. Presentan una gran ventaja al no utilizar gran cantidad de memoria en el dispositivo móvil, además de varios tipos de mensajes que sirven para controlar estados de envío y recepción de información. Esto permite establecer un mecanismo que gestione las interrupciones entre el Front-end y el Back-end.

JADE-LEAP utiliza el servicio de MAS (*Sistema de Gestión Multiagente*) que permite el diseño de la mensajería y RMI (*Invocación de Métodos Remotos*). JADE apoya al manejo de interrupciones intermitentes mediante su protocolo MTP (*Message Transport Protocol*) que lleva a cabo el transporte físico de mensajes entre dos agentes que residen en contenedores distribuidos, brindando servicios de recuperación de fallos. Al darse un quiebre en la comunicación, MTP realiza un monitoreo constante y cuando la reconexión se da a lugar, comunica a los agentes para que reinicien el dialogo desde el punto en que se generó la interrupción.

3.3.1. Proyección de los patrones software en los diagramas de diseño

En el diseño se define la arquitectura de la propuesta que soporte los requisitos. A partir del análisis se puede establecer las clases que forman parte del diagrama de clases del sistema (Figura 3.7-3.11).

Presentamos los diagramas de clase del lado del cliente, que lo llamaremos *Agente Proxy Cliente* (*APC*) y son: aplicación, proxy, proxy-comportamiento. Los diagramas de clase del lado de servidor, que lo llamaremos *Agente Proxy Servidor* (APS) y son: proxy y proxy-comportamiento.

A continuación se explica la relación entre el modelo de patrones software y las clases que se han sido definidas en el diagrama de clases:

- Modelo y patrón *Observer*: Las clases que permiten el almacenamiento temporal y el observador que analiza constantemente el estado del buffer son: *BuscadorVideo* y *SharedPreferences*.

- Vista y patrón *Composite*: Las clases que son ejecutadas para lograr el despliegue del video streaming en el dispositivo cliente son: *VideoView*, *VideoPlayer*, *Uri*, y *MediaController*.

- Controlador: En este se define el patrón *Proxy* el cual está representado por las clases *JadeAplicacion*, *JadeGateway*, *AgenteProxy* y *GatewayAgent*. El patrón *Strategy* también fortalece la capa de control implementado mediante las clases: *ThreadedBehaviourFactory*, *Behaviour* y *CyclicBehaviour*.

El observador externo que permite evaluar el estado del canal de comunicación está representado por las clases: *Socket*, *ConnectionListener*, *ListenerPuertoUDP*, *ListenerPuertoRTSP*, *ListenerMensajeUDP*, *ListenerMensajeRTSP*, *MessageTemplate* y *ACLMessage*.

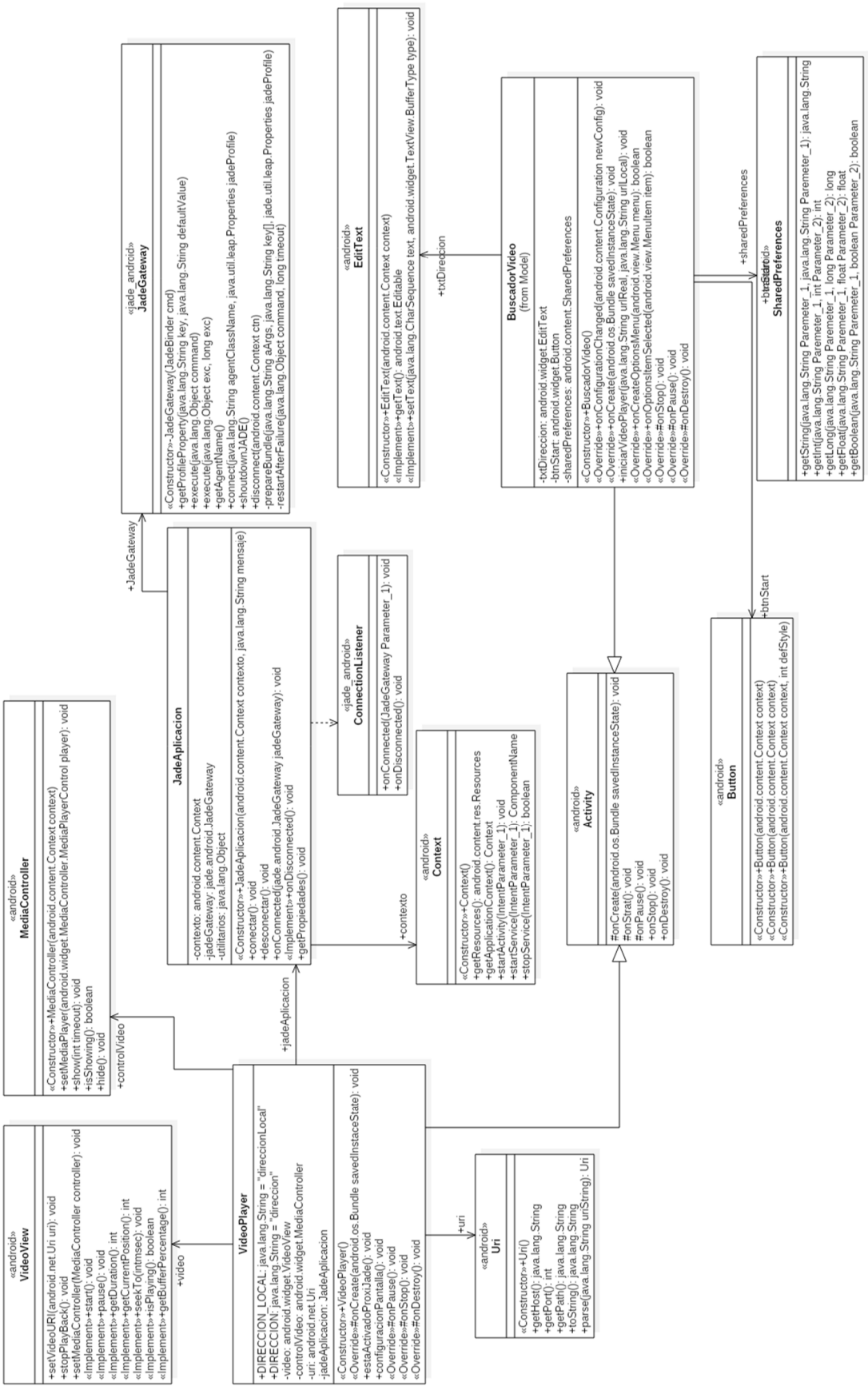


Figura 3.7: Diagrama de clase de APC - Aplicación

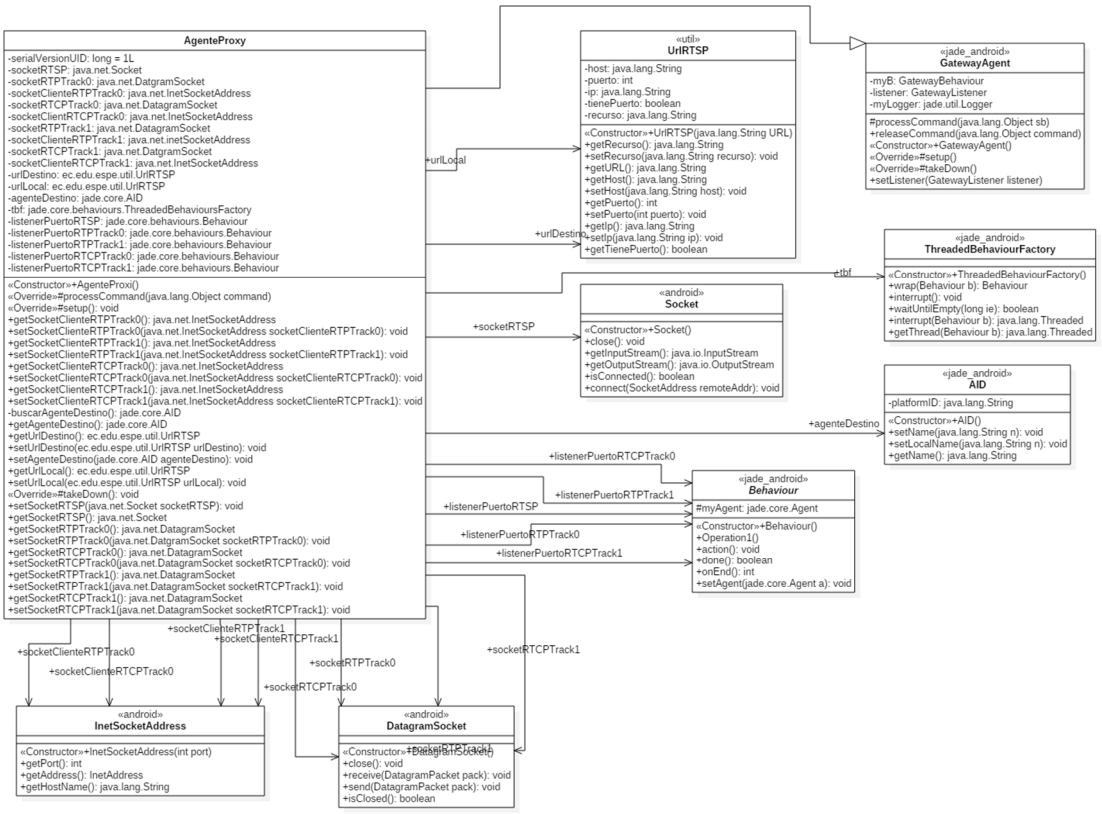


Figura 3.8: Diagrama de clase APC – Proxy

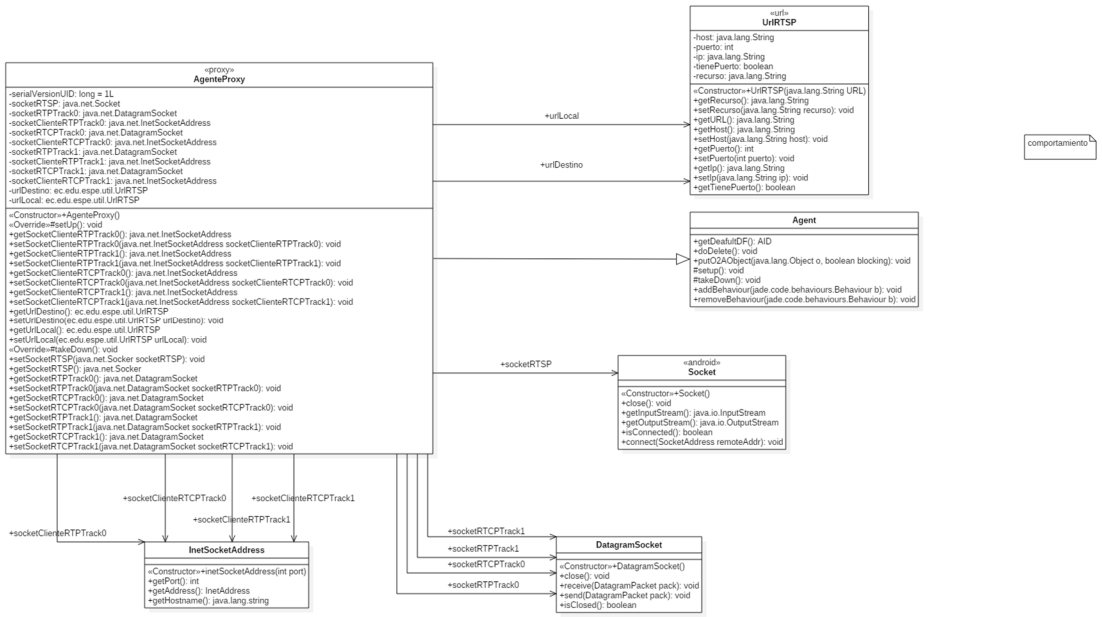


Figura 3.9: Diagrama de clase APS - Proxy

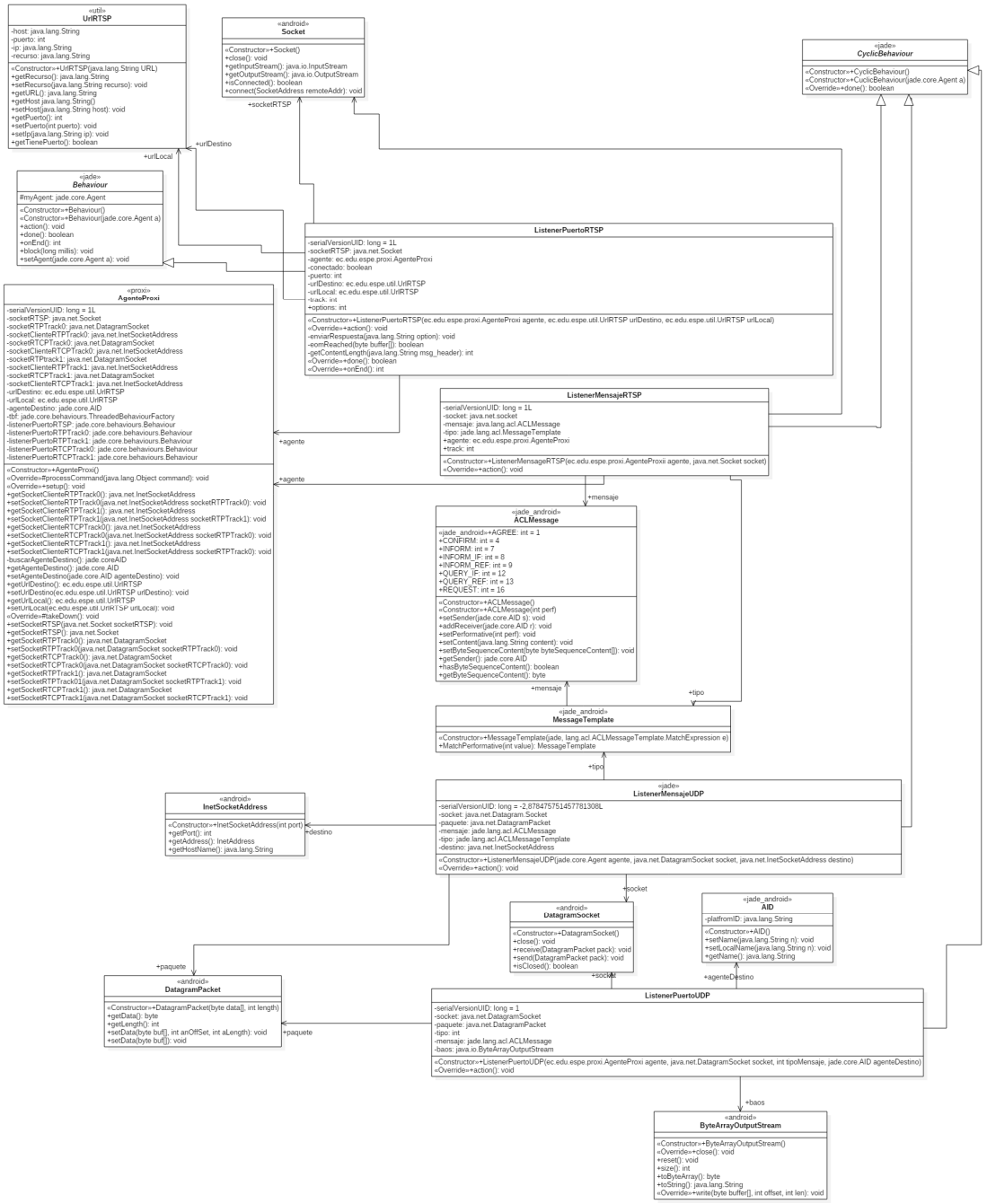


Figura 3.10: Diagrama de clase de APC - Proxy Comportamiento

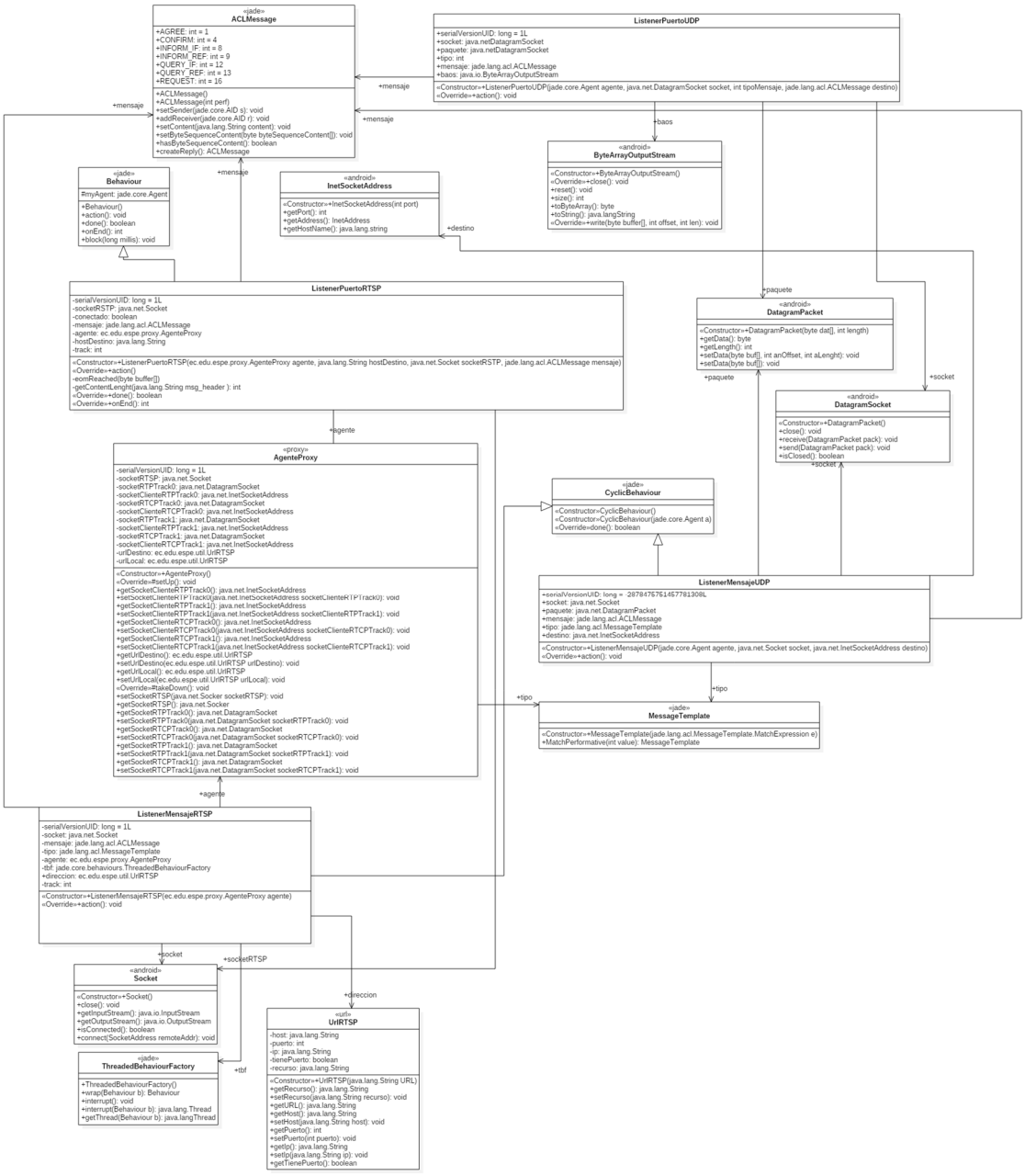


Figura 3.11: Diagrama de clase de APS - Proxy Comportamiento

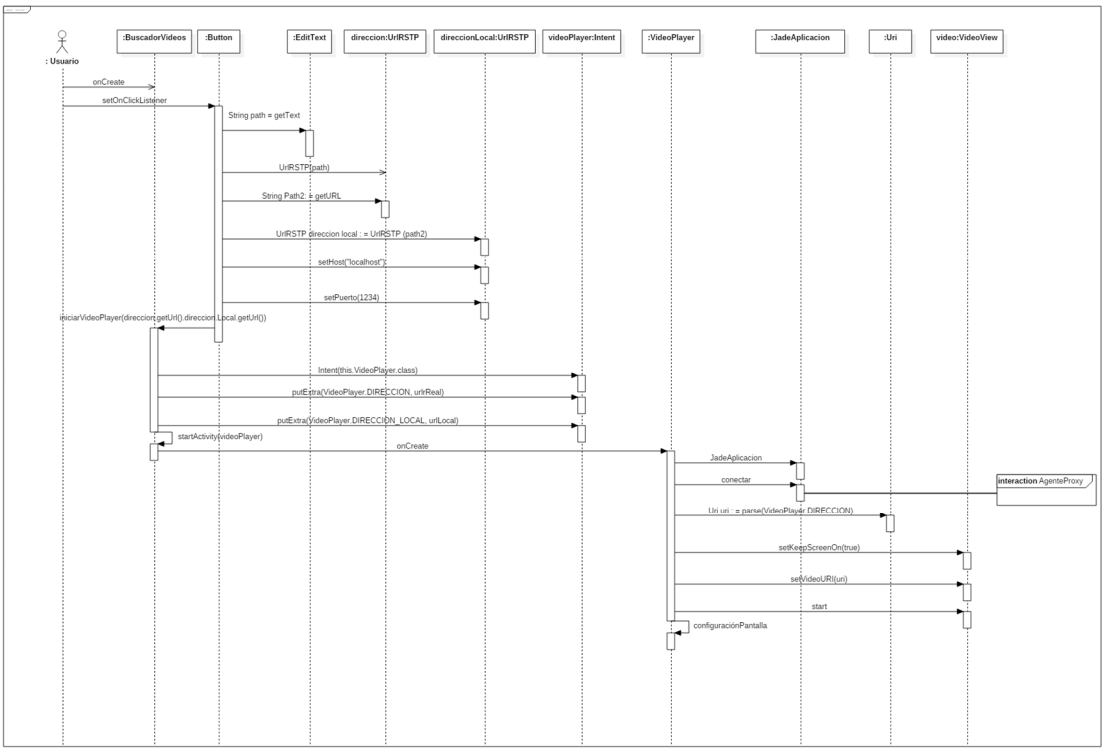


Figura 3.12: Diagrama de secuencia de APC - Reproducir Video - Parte 1

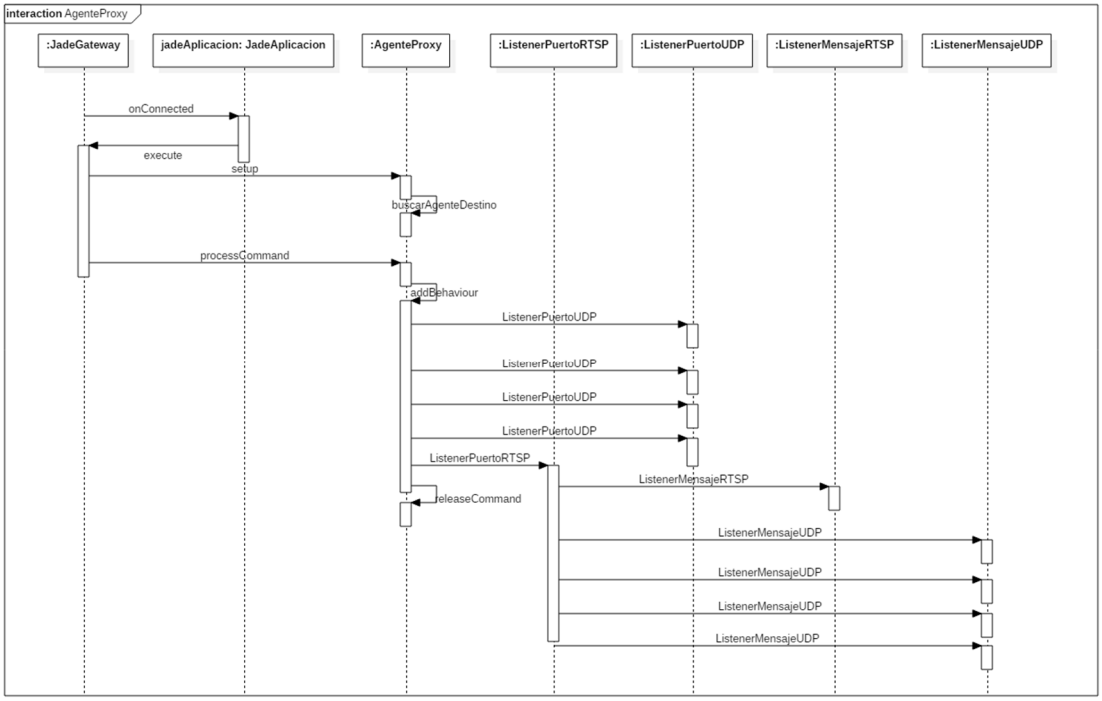


Figura 3.13: Diagrama de secuencia de APC - Reproducir Video - Parte 2

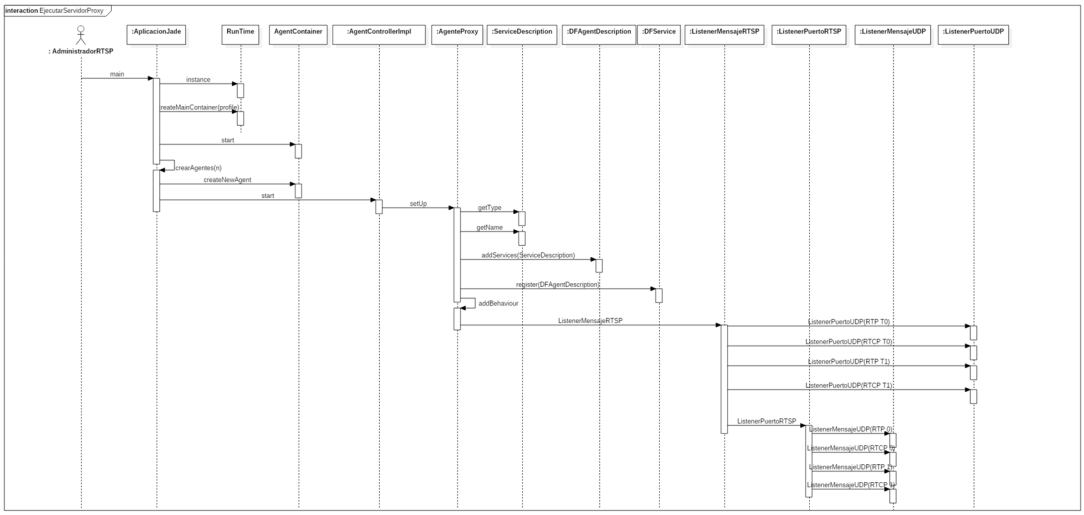


Figura 3.14: Diagrama de secuencia de APS - Ejecutar Servidor Proxy

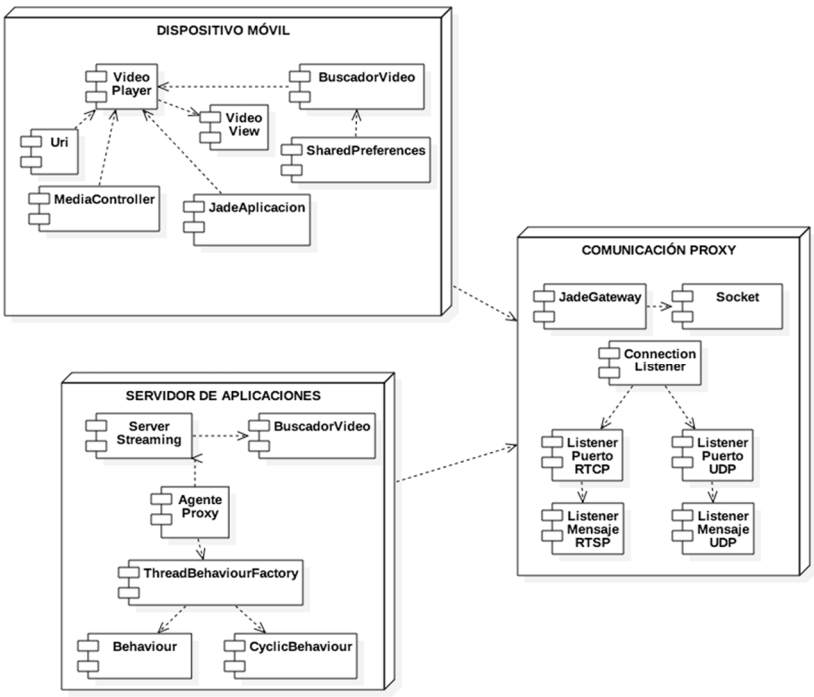


Figura 3.15: Diagrama de despliegue

Los diagramas de secuencia del APC contiene la parte 1 y 2 que corresponde a reproducir video y el diagrama de secuencia del APS contiene ejecutar servidor proxy (Figura 3.12-3.14). Se explica a detalle la comunicación que se da entre las clases para cumplir el establecimiento de la sesión de video streaming, la activación de los

observadores: del almacenamiento y del canal de comunicación, la definición de una interrupción y la selección del algoritmo que se ejecutara de acuerdo al evento ocurrido.

Finalmente tenemos el diagrama de despliegue de la solución base para control de interrupciones y contiene dos componentes que son el servidor de aplicaciones y dispositivo inalámbrico. Cada uno de ellos tiene distribuida las clases que conforman la solución y están distribuidas como se puede observar en la Figura 3.15.

Al iniciar esta visualización el *Proxy* solicita el trabajo de observador a *SharedPreference* quien continuamente está revisando a *BuscadorVideo* y define alerta de umbral superado. Si este evento ocurre el *Proxy* solicita el servicio de *ConnectionListener* de observar el estado del canal de comunicación, si esta respuesta asegura que hay una interrupción el *Proxy* invoca al servicio del *ThreadBehaviourFactory* para que seleccione el algoritmo a ejecutar, en este caso *Behaviour*.

Este algoritmo se encarga de que el *Proxy* solicite al *ServerStreaming* el almacenamiento temporal de los flujos que no pueden ser transmitidos y al *ConnectionListener* pide que evalúe el canal y determine su restablecimiento. Al restablecerse el canal el *Proxy* recibe esta notificación y pide al *ThreadBehaviourFactory* la ejecución del algoritmo correspondiente, para esta situación será *CyclicBehaviour* mediante el cual el *Proxy* solicita la transmisión del flujo que se encuentra almacenado en el *Buffer* del Servidor de Aplicación y sea entregado al proceso de visualización ubicado en el dispositivo móvil.

3.3.2. Desarrollo del modelo con software libre

Se realizó un primer modelo para implantar la arquitectura para paso de imágenes con agente JADE de la siguiente manera (Figura 3.16): un Agente Jade que realiza la función de servidor, al cual se le designó el comportamiento de obtener la imagen del repositorio y enviarla al cliente. En el cliente, que es el dispositivo móvil Android, se instaló otro Agente cuyo comportamiento fue recibir la fotografía y desplegarla en la pantalla.

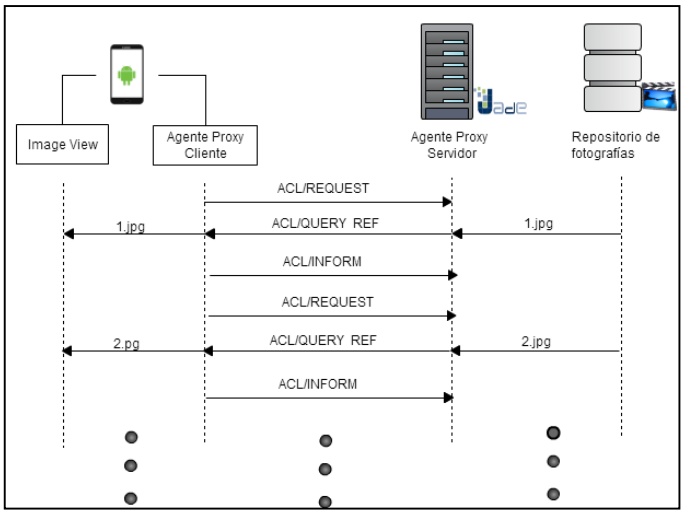


Figura 3.16: Arquitectura para envío de imágenes con agente JADE

Posterior se realizó un segundo modelo que permitió adaptar la arquitectura ya explicada de los agentes Jade al protocolo RTSP/RTP que ha logrado que la técnica de video streaming sea muy utilizada, y esté vigente hasta la actualidad. El éxito del protocolo RTSP, se debe a su velocidad y eficiencia a la hora de transmitir videos de gran tamaño y calidad. A esta eficiencia en los dispositivos inalámbricos se le añade los CODEC de video H.263 y H.264, que comprimen mediante algoritmos el tamaño de los fotogramas, para evitar ocupar un gran espacio de memoria en el dispositivo móvil. RTSP envía al video por dos canales, uno donde emitirá los fotogramas y otra el audio, los cuales obedecen a un instante en el tiempo para poderlos sincronizar.

La característica principal de RTSP es enviar paquetes RTP a través del protocolo de transporte UDP, lo cual hace que los paquetes sean enviados y recibidos de manera rápida, porque no es orientado a la conexión, siendo esta su gran fortaleza, pero también se convierte en su mayor debilidad, ya que nadie garantiza que el cliente reciba los paquetes o que exista la perdida de estos, que es evidente en una interrupción del cliente. Para superar este gran inconveniente, el mecanismo que realiza la transferencia de video en tiempo real sobre teléfonos móviles, determina las siguientes entidades (Figura 3.17): Un Servidor de video RTSP basado en VLC, un APS, un dispositivo móvil que actúa como cliente, albergando un APC y a la aplicación RTSP Cliente la cual se llama PV Player y viene integrada al dispositivo Android, esta generará las peticiones y visualiza el video.

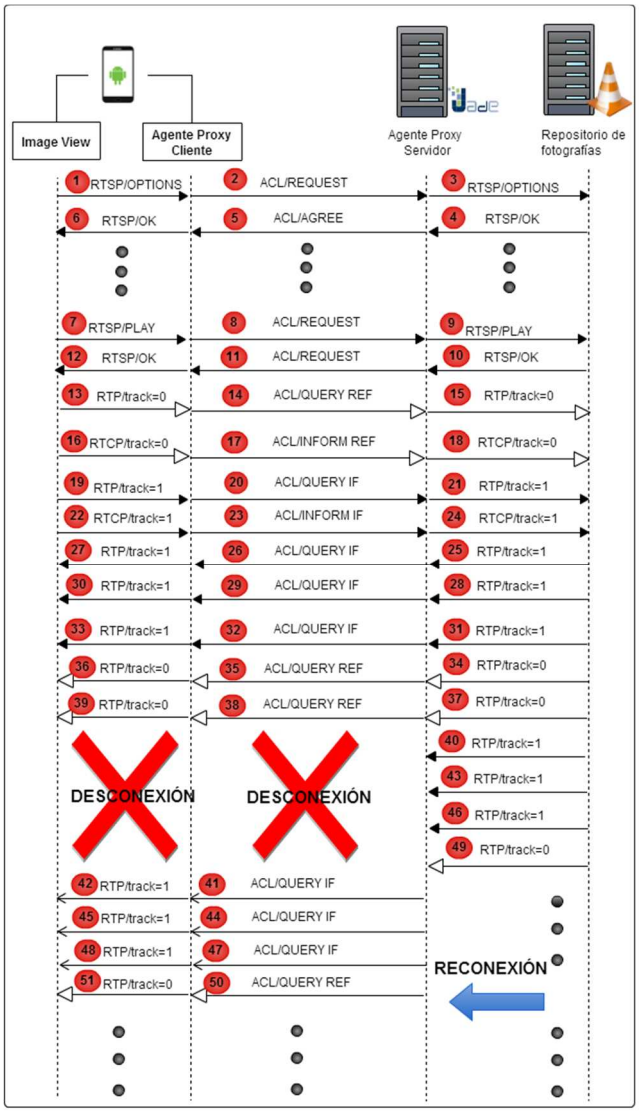


Figura 3.17: Arquitectura JADE - RTSP

Al colocar agentes de software JADE en el APS y APC, éstos cooperan en definir cuándo el móvil está fuera de cobertura al usar señalización MTP, encargándose de resolver las interrupciones intermitentes y reanudar automáticamente la sesión de video streaming. Los mensajes transmitidos FIPA-ACL al teléfono móvil son guardados en el buffer del APS ya que no pueden ser entregados al destinatario, reenviándose una vez que se reanude la conexión.

El servidor de video streaming VLC se conecta con el APS y mediante acceso inalámbrico con el APC. En las comunicaciones establecidas entre los pares servidor–APS y APC–Cliente se transmiten mensajes utilizando el protocolo RTSP y para transmitir video el protocolo RTP. La comunicación entre el par APS-APC, que se despliegan en la

plataforma JADE, se comunican a través de mensajes FIPA ACL, simplificando la comunicación entre agentes.

El APC se divide en dos partes: el Front-End que actúa con el dispositivo móvil Android y el Back-End con el servidor de video streaming. La comunicación entre estas partes es mediante un canal de comunicación inalámbrica, donde las posibles interrupciones se gestionan de forma transparente por el APC.

Además se implementa de forma transparente dos funciones: el mecanismo de almacenamiento y envío, y el intercambio de mensajes de negociación RTSP y los paquetes RTP y RTCP. Con este se garantiza que el agente de dispositivo móvil nunca se va a desconectar del cliente.

El APS recepta los mensajes del APC y los envía al servidor según los puertos establecidos en la negociación y recepta los mensajes provenientes del servidor para renviarlos al APC

El uso de señalización MTP entre los agentes APS-APC permite definir si el dispositivo móvil está fuera de cobertura en una comunicación inalámbrica y reanudar automáticamente la sesión de video streaming. Al ocurrir una interrupción, los mensajes FIPA-ACL ya no se transmiten al dispositivo móvil y son almacenados en el buffer del APS, para ser entregados al APC una vez restablecida la conexión.

Estos mensajes almacenados en el APS son encapsulados y tratados por el servicio de reparto persistente hasta que el agente APC se conecte y puedan ser enviados. Con este proceso se hace transparente el comportamiento de interrupción de la red inalámbrica al servidor y al cliente.

El protocolo MTP permite definir el tiempo de espera para intentar reconectar APS-APC, cuyo valor que por defecto es un minuto. Se debe considerar éste tiempo para el dimensionamiento del buffer.

La mitigación de disrupciones mediante la arquitectura planteada, usando agente de software como JADE permite una alternativa para controlar y mejorar el proceso de transmisión de datos, incrementado de esta forma, la usabilidad del sistema.

Agente proxy cliente

El APC permite que el dispositivo móvil pueda recibir y enviar los mensajes RTSP, los paquetes RTP y RTCP de manera segura. Este APC implementa un mecanismo de almacenamiento y envío, así como el de filtrar la negociación RTSP para que sea lo más transparente posible. Este reside en el dispositivo móvil lo cual asegura que el agente nunca se va a desconectar del cliente. Este agente tiene diez comportamientos y se los presenta en la tabla 3.9:

Tabla 3.9: Comportamiento APC

Comportamiento	Mensaje ACL	Origen	Destino	Descripción
ListenerMensajeRTSP	REQUEST	APC	Servidor VLC Puerto 1234	Negociación RTSP
ListenerMensajeRTPTrack0	QUERY_REF	APC	Servidor VLC Puerto aleatorio	Pista de Audio
ListenerMensajeRTCPTrack0	INFORM_REF	APC	Servidor VLC Puerto aleatorio	Control de Pistas de Audio
ListenerMensajeRTPTrack1	QUERY_IF	APC	Servidor VLC Puerto aleatorio	Pista de Video
ListenerMensajeRTCPTrack1	INFORM_IF	APC	Servidor VLC Puerto aleatorio	Control de Pistas de Video
ListenerPuertoRTSP	AGREE	Servidor VLC Puerto 1234	APC	Respuesta a negociación RTSP
ListenerPuertoRTPTrack0	QUERY_REF	Servidor VLC Puerto aleatorio	APC	Pista de Audio
ListenerPuertoRTCPTrack0	INFORM_REF	Servidor VLC Puerto aleatorio	APC	Control de Pistas de Audio
ListenerPuertoRTPTrack1	QUERY_IF	Servidor VLC Puerto aleatorio	APC	Pista de Video
ListenerPuertoRTCPTrack1	INFORM_IF	Servidor VLC Puerto aleatorio	APC	Control de Pistas de Video

Agente proxy servidor

El APS cumple dos funciones, la primera es la de receptar los mensajes del APC y los envía al servidor según los puertos asignados en la negociación y la función restante es receptar los mensajes provenientes del servidor y enviarlos al APC.

Este agente tiene diez comportamientos y se los presenta en la tabla 3.10:

Tabla 3.10: Comportamiento APS

Comportamiento	Mensaje ACL	Origen	Destino	Descripción
ListenerMensajeRTSP	AGREE	APS	PV Player puerto 1234	Respuesta a negociación RTSP
ListenerMensajeRTPTrack0	QUERY_REF	APS	PV Player puerto aleatorio	Pista de Audio
ListenerMensajeRTCPTrack0	INFORM_REF	APS	PV Player puerto aleatorio	Control de Pistas de Audio
ListenerMensajeRTPTrack1	QUERY_IF	APS	PV Player puerto aleatorio	Pista de Video
ListenerMensajeRTCPTrack1	INFORM_IF	APS	PV Player puerto aleatorio	Control de Pistas de Video
ListenerPuertoRTSP	REQUEST	PV Player puerto 1234	APS	Petición RTSP
ListenerPuertoRTPTrack0	QUERY_REF	PV Player puerto aleatorio	APS	Pista de Audio
ListenerPuertoRTCPTrack0	INFORM_REF	PV Player puerto aleatorio	APS	Control de Pistas de Audio
ListenerPuertoRTPTrack1	QUERY_IF	PV Player puerto aleatorio	APS	Pista de Video
ListenerPuertoRTCPTrack1	INFORM_IF	PV Player puerto aleatorio	APS	Control de Pistas de Video

3.3.3. Resultados experimentales

Para comprobar el potencial de la aplicación desarrollada mediante JADE, este debe ser sometido a una serie de pruebas. La primera se relaciona con la idea más básica de generar y enviar un video, la cual consiste en una secuencia de fotografías debidamente ordenadas. Entonces el primer reto de JADE fue enviar desde el servidor al cliente una serie de imágenes, asegurando que lleguen en el mismo orden y sin ninguna pérdida después de una interrupción. La arquitectura a establecer es la siguiente: un Agente Jade que realiza la función de servidor, al cual se le designó el comportamiento de obtener la imagen del repositorio y enviarla al cliente. En el cliente, que es el dispositivo móvil Android, se instaló otro Agente cuyo comportamiento fue recibir la fotografía y desplegarla en la pantalla.

Una vez que se ha realizado la primera prueba, cuyos resultados son detallados a continuación, se procedió a adaptar este mecanismo al protocolo RTSP/RTP que ha logrado que la técnica de video streaming sea muy utilizada, y esté vigente hasta la actualidad. El éxito del protocolo RTSP, se debe a su velocidad y eficiencia a la hora de transmitir videos de gran tamaño y calidad.

A esta eficiencia en los dispositivos inalámbricos se le añade los CODEC de video H.263 y H.264, que comprimen mediante algoritmos el tamaño de los fotogramas, para evitar ocupar un gran espacio de memoria en el dispositivo móvil. RTSP envía al video por dos canales, una donde emitirá los fotogramas y otra el audio, los cuales obedecen a un instante en el tiempo para poderlos sincronizar.

La característica principal de RTSP es enviar paquetes RTP a través del protocolo de transporte UDP, lo cual hace que los paquetes sean enviados y recibidos de manera rápida, ya que no es orientado a la conexión, siendo así su gran fortaleza, pero se convierte también en su mayor debilidad, ya que nadie garantiza que el cliente reciba los paquetes o que exista la perdida de estos, que es evidente en una interrupción del cliente.

Para superar este gran inconveniente, el mecanismo que realiza la transferencia de video en tiempo real sobre teléfonos móviles, determina las siguientes entidades: Un

Servidor de video RTSP, que actúa como servidor, albergando un Agente Proxy Servidor - APS, un dispositivo móvil que actúa como cliente, albergando un Agente Proxy Cliente - APC y a la aplicación RTSP Cliente la cual se llama PV Player y viene integrada al dispositivo Android, esta genera las peticiones y visualiza el video.

El servidor RTSP que se ha usado para las pruebas es VLC, el mismo que tiene como objetivo emitir el video codificado en un formato compatible con el dispositivo móvil Android. Se ha decidido utilizar la siguiente configuración: para video el CODEC H.263, tasa de bits de 400 kb/s, 25 fps, ancho=176, alto=144 y el para el audio el CODEC AAC, tasa de bits de 128 kb/s, 1 canal y la tasa de muestreo de 44100.

Para realizar las pruebas se ha utilizado los siguientes elementos:

- Una red inalámbrica con tecnología WiFi con un punto de acceso de 54Mbps.
- Un equipo portátil, procesador AMD de 2.20GHz, 4 GB de RAM y tarjeta inalámbrica Atheros AR5009 IEEE 802.11 a/g/n WiFi Adapter; en el cual se ejecuta el Agente Proxy Servidor APS.
- Un dispositivo móvil Android, modelo Nexus One, Procesador Qualcomm QSD 8250 1 GHz, 512MB RAM, Tarjeta microSD de 4GB expandible hasta 32 GB, IEEE 802.11b/g/n; en el cual se ejecuta el Agente Proxy Cliente APC.

Antes de realizar las pruebas, verificamos el paso mensajes ACL entre los agentes *Proxy*, con el sniffer propio de la plataforma JADE, teniendo un resultado exitoso, ya que los mensajes se pasan de manera correcta, sin errores, como se puede ver en la Figura 3.18.

Además se instaló una herramienta sniffer WireShark, que es la que facilita obtener los paquetes de audio y video con toda su información como tiempo, bytes, timestamp, entre otros. El sniffer está ubicado en el cliente móvil Nexus One. Para que esta aplicación funcione se debió dar permisos de root.

El servidor RTSP que se ha usado es el VLC y se ha decidido utilizar la configuración para emitir el video y audio indicada en las tablas 3.11 y 3.12.

Figura 3.18: Paso de mensajes FIPA-ACL

Tabla 3.11: Configuración para emitir video

CODEC	Tasa de Bits	fps	Ancho	Alto
H263	600 kb/s	25	176	144

Tabla 3.12: Configuración para emitir audio

CODEC	Tasa de Bits	Canales	Tasa de muestreo
AAC	128 kb/s	1	44100

Para poder observar las ventajas o desventajas del software desarrollado se ha tomado dos escenarios:

- Escenario 1: con proxy (Jade-Video streaming Versión 1.1).
- Escenario 2: sin proxy.

En ambos escenarios se ha emitido un clip de video (.avi), con una duración de 03:19 minutos, así como también se calculará una muestra de los paquetes de audio y otra de video, para cada software. Para calcular la muestra en los dos escenarios el único valor que cambia es N, que es el tamaño total de paquetes recibidos. Los resultados se muestran en las tablas .3.13 y 3.14.

Fórmula a utilizar:

$$n = \frac{Z^2 pq\, N}{NE^2 + Z^2 pq}$$

Donde:

Z = 0,98
p = 0,50
q = 0,5
E = 0,02

Tabla 3.13: Escenario 1 – con proxy

	N (# total de Paquetes Recibidos)	**n(muestra)**
Audio	8542	560
Video	12072	571

Tabla 3.14: Escenario 2 – sin proxy

	N (# total de Paquetes Recibidos)	**n(muestra)**
Audio	8368	560
Video	11823	571

Se realizó la experimentación y toma de la muestra para los dos escenarios indicados, para valorar los parámetros de jitter y retraso.

Jitter

La Figura 3.19 presenta el jitter para los paquetes de audio.

La Figura 3.20 presenta el jitter para los paquetes de video

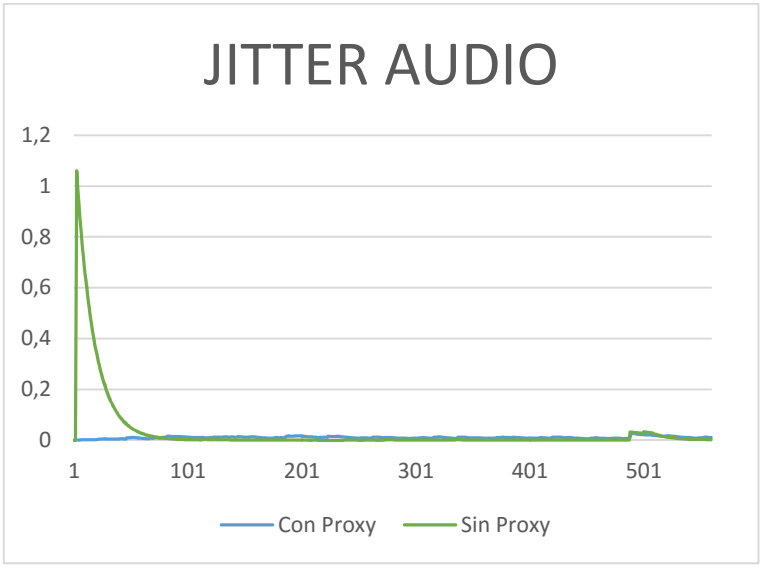


Figura 3.19: Jitter de paquetes de audio

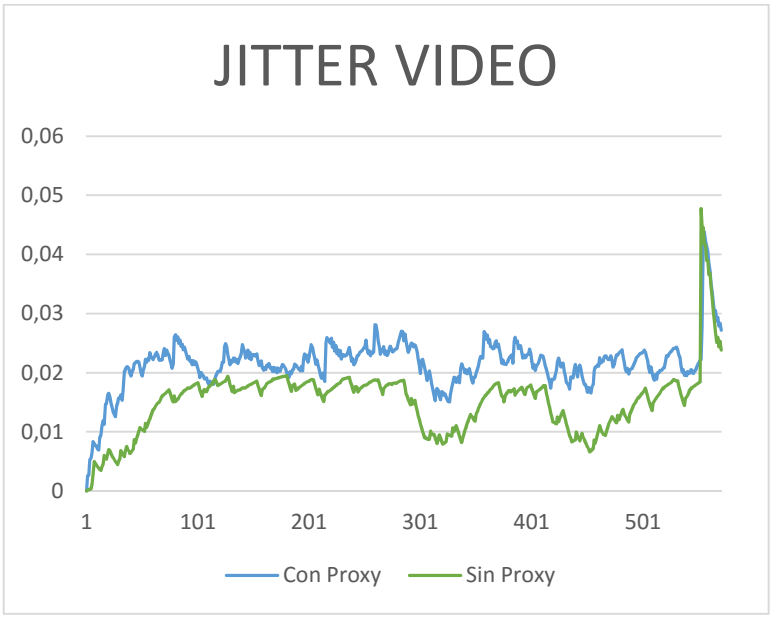


Figura 3.20: Jitter de paquetes de video

Retraso

La Figura 3.21 presenta el retraso para los paquetes de audio

La Figura 3.22 presenta el retraso para los paquetes de video

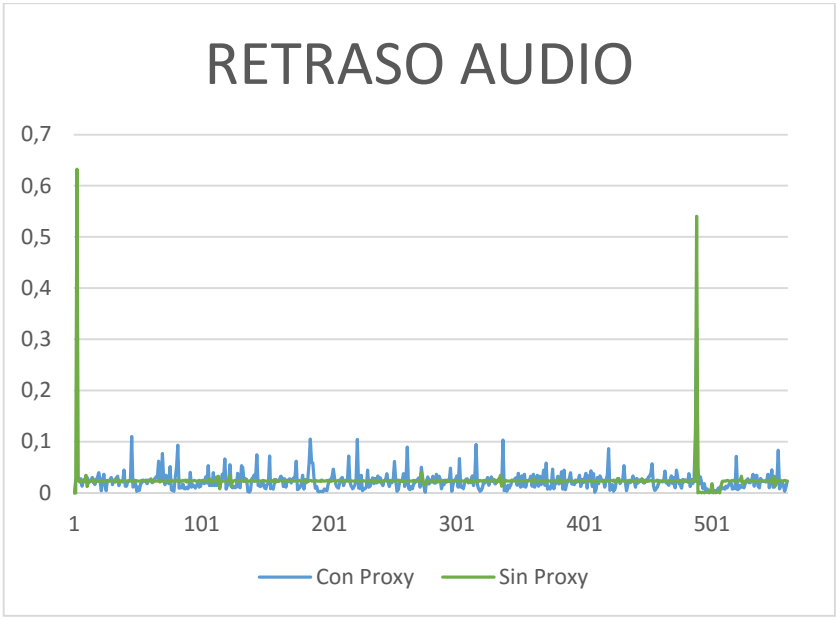


Figura 3.21: Retraso paquetes de audio

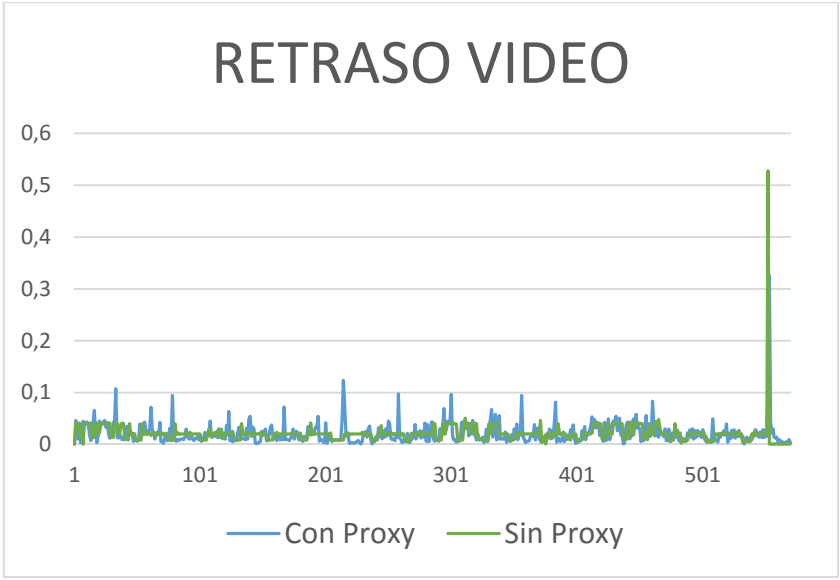


Figura 3.22: Retraso de paquetes de video

Tanto los gráficos del jitter y del retraso nos demuestran que JADE lleva paquetes más grandes, pues poseen mayor información como es la de control, pero si se puede observar mayor estabilidad en la comunicación, ya que no existe interrupciones.

Uno de los beneficios de JADE-ANDROID es que intenta reconectar el APS y el APC durante algún tiempo. El protocolo MTP permite definir el tiempo de espera para que el cliente se reconecte, valor que por defecto es un minuto; éste tiempo influye en el dimensionamiento del buffer. Al restablecerse la conexión entre los agentes, el APC lee los flujos ordenados en el buffer y los envía al video reproductor del Cliente. De esta forma se permite que el cliente recupere el video desde el punto de quiebre.

Los resultados obtenidos es la reproducción de audio y video, con calidad al 100%. Al momento de salir del área de cobertura por aproximadamente 30 segundos, los paquetes tanto de audio y video se recuperan de manera exitosa, sin pérdida de ningún paquete, pero si con una demora al momento de reconectarse, debido a que los paquetes que se han quedado guardados en el buffer deben ser despachados y recalculados su retraso.

En un intervalo de tiempo de 30-45 segundos fuera del área de conexión, los paquetes de audio y video llegan a su destino, el audio se logra escuchar, pero el video se atrasa y muchas veces no se reproduce, ya que el buffer es muy pequeño y no logra recalcular totalmente el flujo.

A un tiempo mayor de 45 segundos, los paquetes de audio y video llegan al APC, pero el PvPlayer no los reproduce. Motivo por el cual, se decidió realizar más pruebas para comprobar la efectividad del sistema, y saber si el software desarrollado estaba fallando o dependía de otros factores ajenos al mismo. El resultado de estas últimas pruebas afirmaron que el motivo del mal funcionamiento de los 30 segundos, se debía al timestamp (marca de tiempo), ya que el PvPlayer decide desplazar los paquetes que tienen un gran retraso y no tomarlos en cuenta, produciendo que la aplicación ya no responda.

Así también cuando existe una interrupción, muchas veces se puede observar que el video se demora un poco más en regresar que el audio, existiendo una desincronización, esto se debe porque un paquete de audio es igual a un flujo, en cambio en el video, un flujo son varios paquetes, teniendo que esperar que lleguen todos los paquetes para que se complete el flujo y poder reproducir. A demás de esto los paquetes

de audio oscilan en un tamaño de 97 a 450 bytes, en cambio los de video de 300 a 1400 bytes, haciendo que el audio utilice menos ancho de banda que el video.

Luego de estas pruebas se ha realizado con el PvPlayer el video streaming sin proxy y a los 30 segundos se recupera de manera inmediata la transmisión, pero se pierden todos los paquetes que se han emitido en el lapso de tiempo de la interrupción. Pasado de los 30 segundos la conexión se corta y ya no regresa ni el audio ni el video.

De los resultados obtenidos se concluye que para obtener una mayor calidad de video streaming, es necesario que:

- El proveedor de servicios de video streaming debe en lo posible realizar sus emisiones de video en tiempo real.
- Configurar de manera correcta todos los parámetros del servidor RTSP.
- Utilizar los CODEC adecuados para cada dispositivo móvil.
- El usuario final se encuentre en un área de cobertura donde la intensidad de la señal sea buena.

Al momento de realizar las pruebas con el software Jade –Video streaming versión 0.9, se utilizó la herramienta wireshark en donde pudimos observar que todos los paquetes se estaban transmitiendo con un tamaño de 1500 bytes, lo cual hizo que se revise el archivo de captura de tráfico de red, y pudimos constatar que la mayoría de los paquetes se completaban con ceros, muchas veces casi vacíos pero su tamaño era constante, es decir 1500 bytes.

Luego de obtener estos resultados, se revisó el APS, donde se verificó el valor real del paquete, que indicaba que siempre estaba enviando con 1500 bytes, valor con el que se inicializó, como se puede ver en la Figura 3.23.

EL cambio que se realizó se muestra en la Figura 3.24, en donde se utilizó un manejador de Bytes (ByteArrayOutputStream), para poder obtener el contenido útil del array y de esta manera enviar en el mensaje JADE.

```
@Override
public void action() {
    byte[] contenido=new byte[1500];
    this.paquete=new DatagramPacket(contenido, contenido.length);
    try {
        socket.receive(paquete);
        mensaje.setPerformative(tipo);
        mensaje.setByteSequenceContent(paquete.getData());
        myAgent.send(mensaje);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figura 3.23: Error de código

```
@Override
public void action() {
    byte[] contenido=new byte[1500];
    this.paquete=new DatagramPacket(contenido, contenido.length);
    try {
        socket.receive(paquete);
        mensaje.setPerformative(tipo);
        baos=new ByteArrayOutputStream();
        baos.write(paquete.getData(), 0, paquete.getLength());
        mensaje.setByteSequenceContent(baos.toByteArray());
        myAgent.send(mensaje);
        baos.reset();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figura 3.24: Corrección en el código

Luego de esta implementación se obtuvo lo esperado, ahora si los paquetes se están enviando con el tamaño real. Este fue un error de programación que generalmente se comete, pero que fue solucionado con éxito, ya que después de esto el resultado fue el Software Video streaming – Jade versión 1.1, en donde el video y el audio se pasa sin distorsión alguna.

Con este ajuste, se pudo llegar a arreglar el error de la distorsión de la imagen y el video, ya que estaba utilizando ancho de banda innecesario.

4. Solución basada en realidad aumentada para control de interrupciones

En este capítulo se describe la funcionalidad incrementada al modelo base con el objetivo de proporcionar ambientes de experiencia inmervisa al usuario mediante sensado móvil, que le permitan minimizar el acceso a zonas de interrupción mediante despliegue de información con realidad aumentada y archivos KML para consulta colaborativa.

4.1. Análisis del diseño basado en patrones software

En el capítulo 3 se planteó el modelo base basado en patrones software de diseño, concretamente en proxy, que ha permitido controlar las interrupciones del servicio video streaming ocasionadas por quiebres del canal de comunicación inalámbrico mediante una arquitectura de proxies. Esto ha permitido entregar al usuario continuidad del servicio debido a que una vez detectado la interrupción pide que se almacene los flujos y éstos sean entregados cuando se haya restablecido el canal. Esta implementación puede incurrir en tiempos adicionales de procesamiento e inversión de recursos, por los proxies y almacenamiento temporal respectivamente. Pero son totalmente despreciables debido a la reducción del impacto negativo, ocasionado por las interrupciones del servicio por quiebres del canal de comunicación, por mitigar la retransmisión de flujos de datos, minimizar tiempos de restablecimiento de la sesión y la continuidad del servicio, impactando y mejorando notablemente la QoE del usuario.

Utilizando este modelo, estamos interesados en presentar un modelo incremental en el que se establezcan componentes adicionales que integre características de experiencia inmersiva con sensado móvil y RA Estas aportaciones permiten brindar QoE al usuario con el propósito de minimizar el acceso a zonas de interrupción del canal inalámbrico. Con el mecanismo diseñado en el capítulo 3 y utilizando nuevas tecnologías se planteó la necesidad de ofrecer nuevas funcionalidades al modelo propuesto en el capítulo anterior. Se desarrolló una arquitectura que permita integrar características de experiencia inmersiva con sensado móvil. Esto permitió entregar al usuario información de la intensidad de la señal del dispositivo móvil mediante técnicas de RA, para minimizar el acceso del usuario a zonas de interrupción del canal inalámbrico. Además se generó mapas personalizados online con la información del nivel de RSSI en puntos geográficos, mediante la construcción de archivos KLM/KMZ para compartirlos a través de Internet con las aplicaciones Google Earth y Google Maps. A continuación se enumeran los requisitos funcionales que deben contener el diseño de este modelo, los cuales se han determinado mediante diagramas de casos de uso y la descripción concreta de los mismos. La Figura 4.1 y las Tablas 4.1-4.11 muestran los casos de uso.

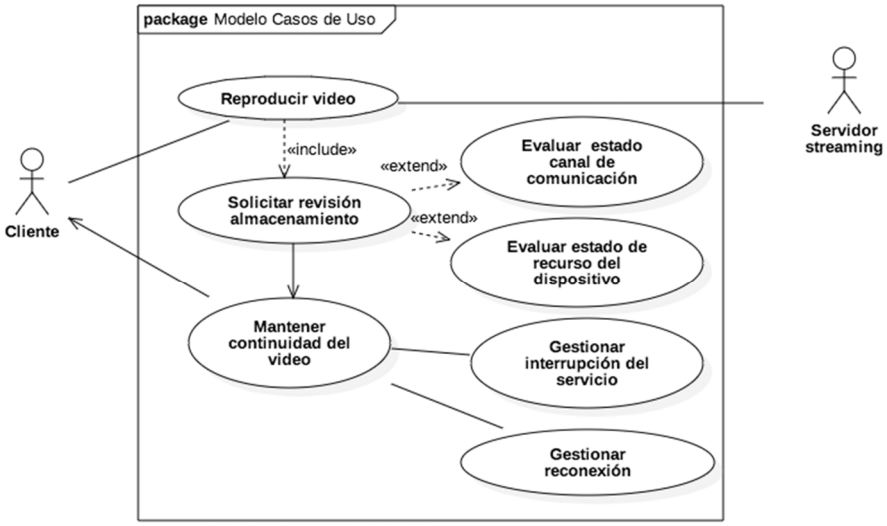


Figura 4.1: Casos de uso modelo basado con RA

El primer caso de uso describe las actividades desde que el cliente solicita el servicio de video streaming hasta que el proxy como intermediario pide al servidor el despliegue en el cliente y su almacenamiento temporal, explicado en Tabla 4.1.

El segundo caso de uso describe la importancia de monitorear el umbral del almacenamiento temporal donde se alojan los flujos, explicado en la Tabla 4.2.

El tercer caso de uso describe toda la funcionalidad referente al análisis por reglas de negocio para evaluar el estado de los recursos del dispositivo, explicado en la Tabla 4.3.

El cuarto caso de uso describe la solicitud de confirmación de una interrupción mediante la evaluación del canal de comunicación, explicado en la Tabla 4.4.

El quinto caso de uso describe el proceso que sigue el proxy para gestionar la continuidad del servicio de video streaming ante una interrupción, explicado en la Tabla 4.5.

El sexto caso de uso describe el uso de buffer para almacenar los flujos que han sido enviados por el servidor y que no pueden ser entregados al dispositivo inalámbrico por la interrupción, explicado en la Tabla 4.6.

El séptimo caso de uso describe la recuperación de los flujos almacenados en el buffer y el despliegue en el cliente, cuando se ha recuperado el canal de comunicación, explicado en la Tabla 4.7.

Tabla 4.1: Casos de uso modelo – reproducir video

Caso de Uso: reproducir video
Actores: cliente, servidor video streaming
Resumen: describe como inicia la petición de reproducción del video y su despliegue.
Precondiciones:
Descripción: 1. El cliente solicita visualización del video. 2. Ingresa los datos IP del servidor, puerto y video. 3. Los datos son validados por el proxy. 4. Si no existen problemas el proxy solicita el servicio de video streaming al servidor. 5. Se despliega una ventana en la que inicia el video streaming. 6. El servidor inicia la entrega de flujos al almacenamiento temporal e inmediatamente los despliega en la ventana.
Pos condiciones: solicita la revisión constante del almacenamiento.
Observaciones:

Tabla 4.2: Casos de uso modelo – revisión almacenamiento

Caso de Uso: solicitar revisión almacenamiento
Actores:
Resumen: explica la importancia de monitorear umbral de almacenamiento temporal
Precondiciones: el proxy solicita la revisión del almacenamiento
Descripción: 5. Se pide constantemente información del umbral del almacenamiento. 6. Se compara con el máximo nivel establecido. 7. Se genera una alerta en caso de superación del umbral definido. 8. Se comunica este problema al proxy
Pos condiciones: solicita la evaluación de los recursos del dispositivo.
Observaciones:

Tabla 4.3: Casos de uso modelo – estado recursos

Caso de Uso: evaluar estado de los recursos del dispositivo
Actores:
Resumen: para proveer de información necesaria para evaluar por regla de negocio el estado de los recursos del dispositivo.
Precondiciones: el proxy solicita la evaluación del estado de los recursos.
Descripción: 1. Se realiza el monitoreo del estado de los recursos. 2. Se efectúa la regla de negocio de los recursos: RSSI y geolocalización y se determina colores a desplegar con información. 3. Se despliega información al usuario.

4. Se genera archivo de información KML/KMZ 5. Se almacena la información.
Pos condiciones: se solicita evaluar estado de canal de comunicación.
Observaciones:

Tabla 4.4: Casos de uso modelo – estado canal

Caso de Uso: evaluar estado canal de comunicación
Actores:
Resumen: para confirmar que se trata de una interrupción, se evalúa el estado del canal de comunicación.
Precondiciones: proxy solicita evaluación del canal de comunicación
Descripción: 4. Se evalúa el canal de comunicación. 5. Se determina el estado del canal de comunicación 6. Este estado es comunicado al proxy.
Pos condiciones: se solicita mantener la continuidad del servicio
Observaciones:

Tabla 4.5: Casos de uso modelo – continuidad video

Caso de Uso: mantener continuidad del video
Actores:
Resumen: dado un evento del canal de comunicación debe gestionarse la continuidad del servicio video streaming.
Precondiciones: el proxy solicita ejecución de proceso de acuerdo al estado del canal de comunicación.
Descripción: Alerta interrupción : 3. En caso de ser una alerta de interrupción se solicita la gestionar interrupción del servicio. Alerta conexión: 4. Cuando el canal se ha restablecido se solicita gestionar reconexión.
Pos condiciones:
Observaciones:

Tabla 4.6: Casos de uso modelo – gestión interrupción

Caso de Uso: gestionar interrupción del servicio
Actores:
Resumen: explica las actividades a cumplir cuando ocurre una interrupción y como se controla la misma.
Precondiciones:
Descripción: 4. Se solicita el almacenamiento de los flujos en el buffer.

5. Se despliega en la interfaz gráfica del usuario el mensaje “Reconectando” por el problema dado y que se encuentra en estado de espera para reconectarse automáticamente 6. Se pide el servicio de evaluar continuamente el canal de comunicación.
Pos condiciones:
Observaciones:

Tabla 4.7: Casos de uso modelo – gestión reconexión

Caso de Uso: gestionar reconexión
Actores:
Resumen: explica las actividades a cumplir cuando el canal se ha restablecido.
Precondiciones:
Descripción: 3. Se solicita que los flujos almacenados en el buffer sean transmitidos al dispositivo cliente. 4. El cliente automáticamente continúa visualizando el video en el punto en el que se encontraba cuando ocurrió la interrupción.
Pos condiciones:
Observaciones:

En la Figura 4.2 se especifica el caso de uso de *evaluar estado de recursos del dispositivo*, a continuación se detalla la funcionalidad de la regla de negocio del estado de los recursos del dispositivo con sus respectivos casos de usos.

El octavo caso de uso describe la importancia de monitorear el estado de los recursos del dispositivo, para informar al usuario que está entrando en zonas de interrupción, explicado en la Tabla 4.8.

El noveno caso de uso describe el pedido de realizar proceso de evaluación mediante regla de negocio del RSSI, explicado en la Tabla 4.9.

El décimo caso de uso describe el análisis por regla de negocio del RSSI, explicado en la Tabla 4.10.

El undécimo caso de uso describe todo el proceso de generación y construcción del mapa online, explicado en la Tabla 4.8.

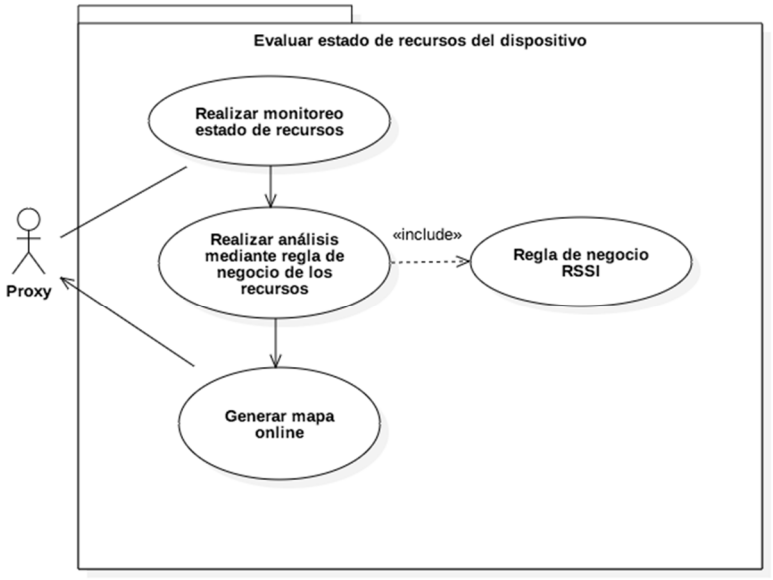


Figura 4.2: Caso de uso regla de negocio

Tabla 4.8: Casos de uso regla de negocio – monitoreo estado recurso

Caso de Uso: realizar monitoreo estado de recursos
Actores: proxy
Resumen: se explica la importancia de monitorear el estado de los recursos para informar al usuario que está entrando en zonas de interrupción.
Precondiciones: proxy solicita la revisión del estado de recursos del dispositivo.
Descripción: 1. Continuamente se obtiene la información del estado a los recursos del dispositivo: RSSI y geolocalización 2. Esta información es almacenada temporalmente. 3. La información de los recursos es monitoreada constantemente. 4. Se envía al proxy la información generada para almacenarla.
Pos condiciones: se solicita análisis por regla de negocio de los recursos.
Observaciones:

Tabla 4.9: Casos de uso regla de negocio – análisis regla de negocio

Caso de Uso: realizar análisis por regla de negocio de los recursos.
Actores:
Resumen: se evalúa el estado de los recursos del dispositivo para entregar información al usuario para evitar que entre a zonas de interrupción
Precondiciones: el proxy solicita el análisis por regla de negocio
Descripción: Curso Normal: 1. Se solicita el análisis por regla de negocio de RSSI.
Pos condiciones: se solicita generar mapa online.

Observaciones:

Tabla 4.10: Casos de uso regla de negocio – regla negocio RSSI

Caso de Uso: análisis por regla de negocio de RSSI
Actores:
Resumen: se realiza el análisis por regla de negocio de RSSI.
Precondiciones:
Descripción: 1. Se revisa el RSSI en dBm. 2. Se ubica punto geográfico de la medida de RSSI. 3. Se establece si el valor es debajo de -30dBm, conexión excelente y asignar color verde para publicar ruta en archivo KML. 4. Asociar resultado al punto geográfico definido por GPS. 5. Se establece que valores entre -30dBm y -40dBm conexión buena y se asigna color naranja para publicar ruta en archivo KML. 6. Asociar resultado al punto geográfico definido por GPS. 7. Se establece que valores por encima de -40dBm, conexión mala y asignar color rojo para publicar ruta en archivo KML. Además informar que presente mensaje “Entrando en zona de interrupción”. 8. Asociar resultado al punto geográfico definido por GPS. 9. Se informa al proxy de cambio color y se Almacenar información en almacenamiento temporal.
Pos condiciones:
Observaciones:
Observaciones:

Tabla 4.11: Casos de uso regla de negocio – generar mapa online

Caso de Uso: generar mapa online
Actores:
Resumen: con la información procesada se genera el archivo KML/KMZ
Precondiciones:
Descripción: Curso Normal: 1. Se obtiene el análisis por regla de negocio de RSSI. 2. Se adiciona nuevos datos para construir mapa online. 3. Se arma y construye archivo KML/KMZ 4. Se envía el archivo al almacenamiento temporal.
Pos condiciones:
Observaciones:

4.1.1. Justificación de los patrones software a utilizar

El modelo base basado en patrones software de diseño se planteó un diseño sustentado en patrones software robustos que apoyaron en establecer mecanismos que permitieron gestionar efectivamente el control de una interrupción, este diseño permitió la organización estructurada del modelo por capas modelo, vista y controlador, se potencio a cada una de ellas con los patrones software *Observer*, *Strategy* y *Composite*. Estamos interesados en mantener el diseño anterior y definir componentes incrementales que posibiliten la experiencia con RA que permita al usuario alertar de un posible ingreso a zona de interrupción.

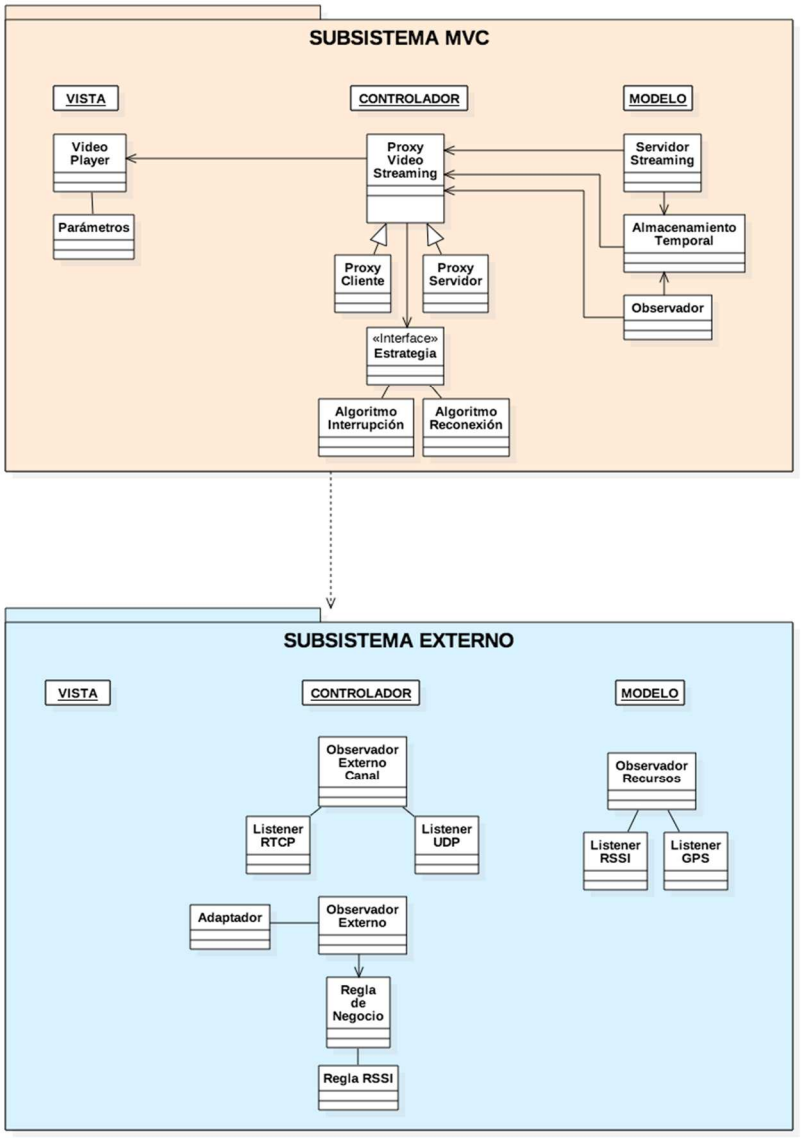


Figura 4.3: Modelo con RA

La Figura 4.3 se ha establecido un incremento funcional en el subsistema externo que permite el monitoreo constante de los recursos del dispositivo móvil y el análisis por regla de negocio para informar al usuario de posible ingreso en zona de interrupción, en este caso se ha utilizado el patrón *Observer*. El patrón *Adapter* convierte el resultado de la regla de negocio en un archivo KML de mapa online.

A continuación se describe el subsistema MVC:

- *Modelo:* se encuentra ubicado en el servidor de aplicaciones. El modelo se fortalece con la implementación del patrón *Observer* que permanentemente está monitoreando el estado del almacenamiento temporal y comunica al *Proxy*.
- *Vista*: se despliega en los dispositivos inalámbricos utilizando el patrón de diseño *Composite*. En caso de ocurrir una interrupción, se mostrará el mensaje *Reconectando...*, el cual permanece activo hasta cuando el canal se restablece y la visualización del video continúa. Además despliega el mensaje *Entrando en zona de interrupción*, cuando se encuentra de el parámetro de RSSI de acuerdo a la regla de negocio lo solicite.
- *Controlador*: está en el servidor de aplicaciones, utiliza los patrones software: *Proxy* y *Strategy*. El patrón *Proxy* es el intermediario entre el servidor de video streaming y el video player. Una de sus actividades principales es enviar la información del recurso de dispositivo móvil, al subsistema externo para que se implemente la regla de negocio correspondiente para informar al usuario con RA y KML. Si se despliega una alerta de canal desconectado por parte del observador externo canal, *Proxy* solicita al patrón *Strategy* la ejecución del algoritmo interrupción. Al momento de restablecerse la comunicación, el *Proxy* se encarga de requerir la ejecución del algoritmo de reconexión.

En el subsistema externo tenemos:

- En el lado del modelo se almacena la información del hardware del dispositivo cliente: RSSI y geolocalización. Se fortalece con la implementación del patrón *Observer* que constantemente está monitoreando el estado de los recursos del dispositivo móvil y se lo comunica al *Proxy*.

- En el lado del controlador el observador externo recibe la información del hardware del dispositivo móvil remitido por el *Proxy*; realiza el análisis por regla de negocio y construye el archivo con esta información. Informa al *Proxy* para que actualice el cambio de posición y color del ícono de RSSI para visualización del usuario con RA. Entrega al patrón *Adapter* el archivo para convierta en archivo KML consumible con Google Earth y Maps. El patrón *Observer* canal valida una interrupción del canal y alerta al *Proxy* para que gestione la interrupción.

Es importante detallar la funcionalidad del aplicativo con RA:

El video player solicita el despliegue del video al *Proxy*, elevando la petición al servidor de video streaming. Continuamente se registra el estado de los recursos del dispositivo: RSSI y geolocalización. El estado de estos recursos son revisados por el patrón "observador de recursos" del subsistema externo y comunicados al *Proxy*.

El *Proxy* solicita al observador del subsistema externo ejecute la regla de negocio del estado de RSSI. Con esto informa proactivamente al usuario del estado del canal y si posiblemente esta por ingresar a una zona de interrupción. De igual manera *Proxy* solicita estado del canal de comunicación al observador externo canal.

Si se presenta una alerta de interrupción, este evento ocasiona que el *Proxy* solicite al patrón *Strategy* la ejecución del algoritmo interrupción. Esta acción solicita que se almacene los flujos de video en almacenamiento temporal. Despliega en la vista el mensaje de que se encuentra en modo desconectado e inicia el bucle de reconexión.

Cuando el *Proxy* recibe la notificación de que el canal se ha restablecido, solicita al patrón *Strategy* la ejecución del algoritmo de conexión y solicita al buffer la entrega de los flujos almacenados.

4.1.2. Diagrama arquitectónico y de despliegue

En la Figura 4.4 se define la arquitectura propuesta mediante el diagrama de clases.

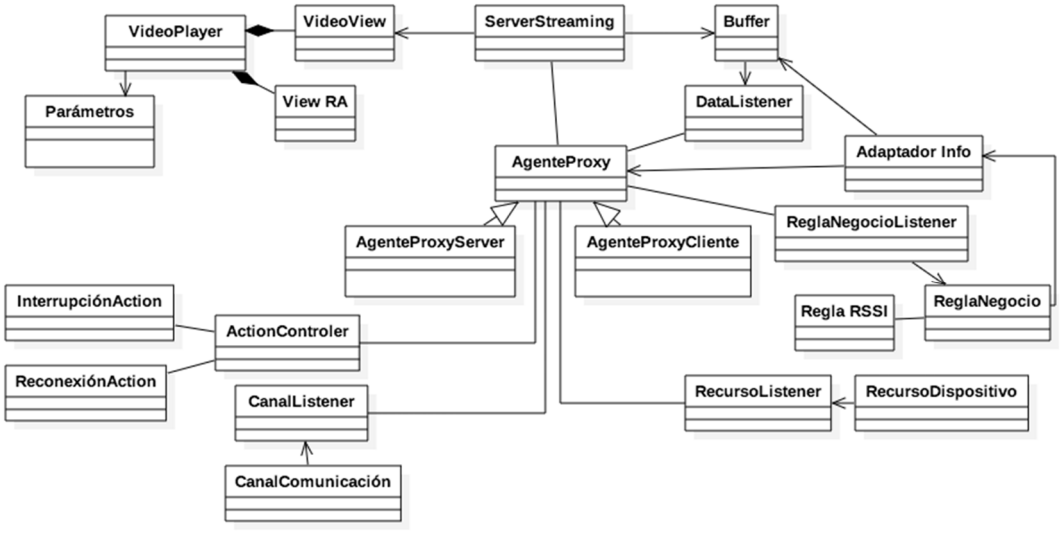


Figura 4.4: Diagrama de clases modelo con RA

- Capa modelo y patrón *Observer* y *Adapter*: engloban las clases que determinan el almacenamiento temporal y monitoreo el estado del umbral del buffer. Estas son: *AgentProxy*, *ServerStreaming*, *DataListener*, *Adaptadorinfo* y *Buffer*.

- Capa vista y patrón *Composite*: contiene las clases que permiten interactuar y desplegar el video con el servidor video streaming. Estas son: *VideoPlayer*, *Parámetros*, *ViewRA* y *VideoView*.

- Capa controlador y patrones software Proxy y *Strategy*: contiene las clases para controlar las interrupciones y tomar acciones. Estas son: *AgenteProxyCliente*, *AgenteProxyServidor*, *ActionControler*, *InterrupciónAction*, *ReconexiónAction*, *ReglaNegocioListener*, *ReglaNegocio* y *ReglaRSSI*.

- El observador externo y patrón *Observer*: contiene las clases que evalúan el estado del canal de comunicación y recursos de dispositivo. Estas son: *CanalListener*, *CanalComunicación*, *RecursoListener* y *RecursoDispositivo*.

La Figura 4.5 describe el diagrama de secuencia del modelo base diseñada, para la interacción entre el cliente y el servidor al solicitar la visualización de un video.

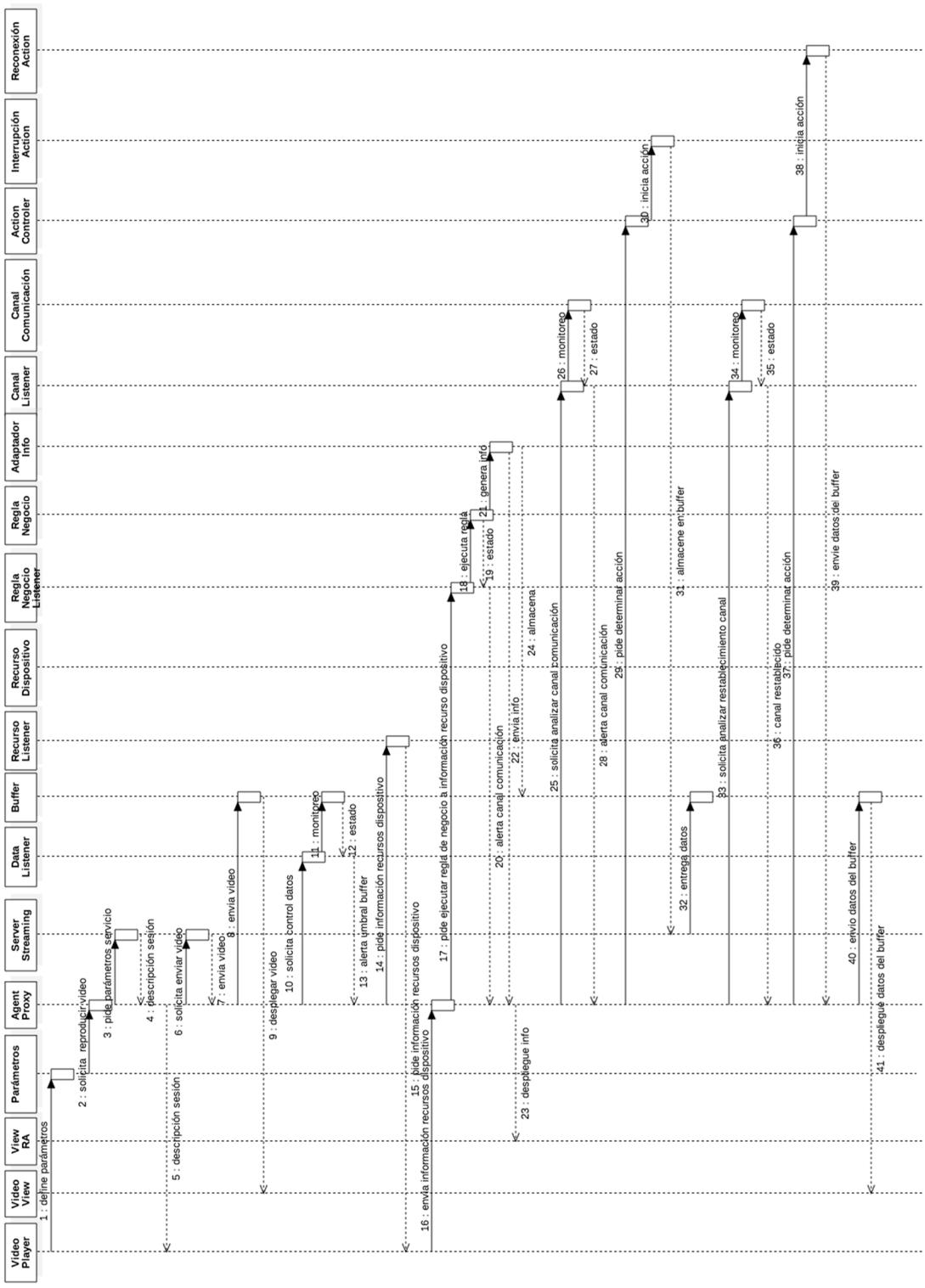


Figura 4.5: Diagrama de secuencia modelo con RA

El *VideoPlayer* es quien inicia la aplicación, definiendo los parámetros mediante la clase *Parámetros*. Posterior solicita reproducir video y eleva la petición a *Proxy* quien valida y pide parámetros de servicio al *ServidorStreaming*. Éste entrega descripción de

sesión al *Proxy* para que los envíe a *VideoPlayer*. Una vez establecida la sesión, *Proxy* solicita a *ServerStreaming* envíe video. *Buffer* recibe el flujo de datos e inmediato entrega a *VideoView* para su despliegue. Por su parte *Proxy* solicita control de datos a *DataListener* sobre estado de umbral del *Buffer*. *Buffer* envía estado a *DataListener* y éste evalúa, determinando si existe alerta. *Proxy* pide información de recursos del dispositivo a *RecursoListener*. *VideoPlayer* envía la información solicitada y ahora *Proxy* pide ejecutar regla de negocio a *ReglaNegocioListener* para determinar umbrales de la RSSI y conforme la regla informar al usuario si se encuentra con buena o mala señal y si están entrando a una zona de interrupción, mediante la publicación de información y mensajes, que le permiten tomar decisiones al usuario proactivamente. Además empieza a generar archivo KML para publicarlo para generar ambiente colaborativo con los usuarios de la red de comunicaciones. Al darse alerta informa a *Proxy* para que confirme mediante monitoreo al canal de comunicación con *CanalListener*. Éste monitorea el canal de comunicación y entrega estado conectado o desconectado a *Proxy*. Si el estado entregado es canal desconectado inmediato *Proxy* solicita acción a *ActionControler*, quien determina la acción a realizarse. Con el estado de canal desconectado *ActionControler* invoca a *InterrupciónAction* para que ejecute acciones. La acción a ejecutarse es indicar al servidor que envíe el flujo de datos al *Buffer* mientras se mantenga es estado desconectado. *Proxy* solicita seguir monitoreando al canal e informe si existe cambio de estado. Al presentarse cambio de estado a canal conectado, vuelve a solicitar acción a *ActionControler*, quien invoca a *ReconexiónAction* para que ejecute acciones. La acción es indicar que Buffer envíe los flujos almacenados para que *VideoView* despliegue el video.

En la Figura 4.6 se observa el diagrama de despliegue, el dispositivo móvil aloja a *VideoPlayer*, *VideoView* y *Parámetros*, que son las clases que inician la solicitud de pedido de servicio video streaming. Además tenemos *ViewRA* que es la clase que me permite desplegar información con RA. Las clases *AgenteProxyCliente* y *AgenteProxyServidor* se comunican mediante mensajes FIPA-ACL y gestionan el establecimiento de la sesión video streaming con el servidor. El buffer del dispositivo móvil recibe los flujos e inmediatamente los entrega a *VideoView* para su despliegue.

Al iniciar la visualización *Proxy* solicita a *DataListener* valore al *Buffer* e informe si existe alerta de umbral superado. *Proxy* solicita al observador de recurso actualice información y pide al observador Regla de Negocio se encargue de evaluar la información y despliegue en la pantalla con RA para que el usuario tome acciones proactivas para evitar ingresar en zonas de interrupción. Si la regla de negocio le indica posible interrupción solicita confirmar con el observador de canal de comunicación la interrupción.

Si esta respuesta asegura que hay una interrupción, el *Proxy* invoca al servicio del *ActionControler* para que seleccione el algoritmo a ejecutar, en este caso *InterrupciónAction*. Este algoritmo se encarga de que el *Proxy* solicite al *ServerStreaming* el almacenamiento temporal de los flujos de video y pide se evalúe el canal y determine su restablecimiento. Al restablecerse el canal, *Proxy* recibe esta notificación y pide al *ActionControler* la ejecución del algoritmo correspondiente, siendo ahora *ReconexiónAction* mediante el cual el *Proxy* solicita envíe los flujos almacenados al buffer del dispositivo móvil para su posterior visualización. Finalmente se entrega el archivo generado KML que indica que lugares tienen buena y mala cobertura.

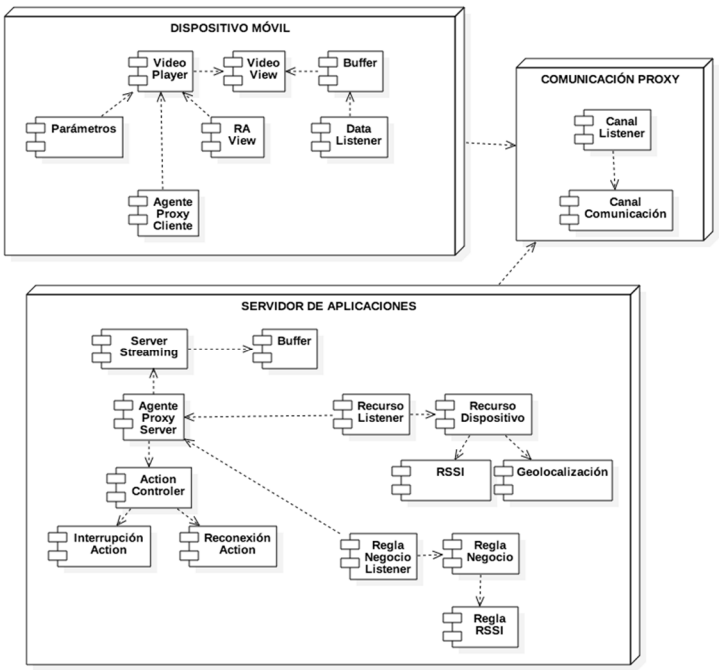


Figura 4.6: Diagrama de secuencia modelo con RA.

4.2. Modelo matemático de rendimiento

Este modelo mantiene la funcionalidad del modelo base mediante el uso de patrones software de diseño, que implementan dos patrones software *Observer*; uno para monitorear el umbral de almacenamiento temporal del dispositivo móvil y otro para validar el estado del canal de comunicación. Con esta implementación se puede controlar la interrupción. Si se presenta una interrupción el mecanismo pide almacenar los flujos emitidos por el servidor en un buffer; consiguiendo así minimizar los tiempos de inicio de sesión. Esto permite que ya no se tenga que empezar de nuevo la visualización del video, sino simplemente pedir la descarga de lo que este en el buffer, una vez que el canal de comunicación este operativo. Se adiciona al modelo un patrón que permite monitorear los recursos del dispositivo e indicar al usuario información que le permita tomar decisiones y cambios de ruta debido a que el mecanismo ahora informa acerca de cómo está su nivel de intensidad de señal en pantalla y además despliega en pantalla cuando está ingresando en zonas de interrupción. Esto permite que el usuario tenga proactividad ante una posible interrupción.

La ecuación del modelo matemático del capítulo 3 se mantiene:

$$T_{EJECUCION} = T_{INICIO_{FLUJO}} + \sum_{i=1}^{n} T_{DATO_i} + \sum_{i=1}^{n} T_{INTERRUPCION_i} + T_{DATOn}$$

Donde:

T_{INICIO} es el tiempo de inicio de solicitud del servicio de video streaming.

$\sum_{i=1}^{n}(T_{DATO_i})$ es el tiempo total de T_{DATO} transmitidos una sola vez.

$\sum_{i=1}^{n} T_{INTERRUPCION_i}$ es el tiempo total de interrupciones.

T_{DATO_n} es tiempo del último T_{DATO} transmitido.

Se realiza la evaluación del modelo matemático planteado con un video de duración de 545 segundos y ejecutado 3 interrupciones, se han realizado pruebas en dos tipos de escenarios: peor y mejor.

Los datos experimentales para el mejor escenario se presentan en la tabla 4.12. Los datos experimentales para el peor escenario se presentan en la tabla 4.13.

Se realizó la gráfica de los escenarios y al compararla con la obtenida en el modelo planteado para el servicio video streaming tenemos las siguientes conclusiones:

- Mejora notablemente el tiempo de ejecución debido a que no tenemos tiempos acumulados por retransmisiones de flujos.
- En este modelo no influye si las interrupciones se dan al inicio o al final del video.
- Se demuestra que el tiempo de ejecución es menor con la utilización del modelo base para control de interrupciones, debido a que no penalizamos con tiempos de retransmisión de flujos.
- Por la proactividad que entrega este modelo al presentar información del comportamiento de la intensidad de la señal del dispositivo, le permite mantenerse en zonas de cobertura. Con esto los escenarios peor y mejor presentan comportamiento similar, como se observa en la gráfica.
- Es importante considerar que el modelo se sustenta en utilizar un almacenamiento temporal en el servidor de aplicaciones, para almacenar los flujos que no se pueden enviar al cliente por interrupción del canal de comunicación. Entonces ahora es necesario analizar el coste que me implicaría y si impacta en el modelo.

Tabla 4.12: Datos experimentales mejor escenario

MEJOR ESCENARIO										
PRUEBA	Tinicio	Tflujo1	Tflujo2	Tflujo3	Tflujo4	Tint1	Tint2	Tint3	Tflujovideo	Tejecucion
1	3,08	14,43	25,3	38,77	466,5	12,28	13,16	10,9	545	584,42
2	2,11	18,8	18,13	10,04	498,03	11,86	11,55	14,83	545	585,35
3	2,21	17,96	13,96	29,26	483,82	11,83	13,41	9,06	545	581,51
4	2,43	11,8	22,18	54,43	456,59	12,21	10,9	16,06	545	586,6
5	2,8	16,32	50,2	34,35	444,13	21,11	9,54	8,98	545	587,43
6	2,76	30,75	28,65	32,79	452,81	11,05	12,23	13,25	545	584,29

Tabla 4.13: Datos experimentales peor escenario

PEOR ESCENARIO										
PRUEBA	Tinicio	Tflujo1	Tflujo2	Tflujo3	Tflujo4	Tint1	Tint2	Tint3	Tflujovideo	Tejecucion
1	2,23	123,34	134,21	225,41	62,04	11,8	12,14	16,38	545	587,55
2	3,18	345,34	115,08	51,98	32,6	12,11	13,31	9,44	545	583,04
3	2,7	76,34	152,16	234,81	81,69	12,69	8,81	15,38	545	584,58
4	2,43	84,34	197,79	223,81	39,06	10,96	11,97	13,84	545	584,2
5	2,28	144,01	147,45	217,51	36,03	12,29	11,63	11,78	545	582,98
6	2,1	88,74	130,88	83,13	242,25	14,99	12,23	18,54	545	592,86

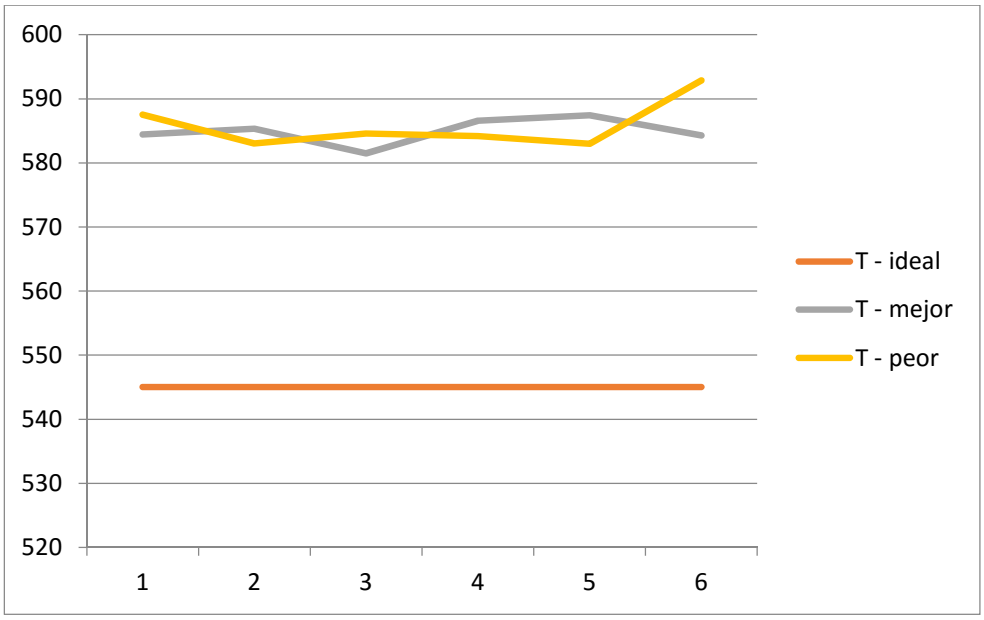


Figura 4.7: Tiempos de ejecución en escenarios

La cantidad de memoria requerida en un escenario con un tiempo de interrupción de 21,1 segundos, que corresponde a la máxima interrupción presentada en el peor escenario experimental, y con una capacidad del canal ADSL de 10Mbps es:

$$Capacidad\ memoria = Capacidad\ canal \times Tiempor\ interrupción$$

$$Capacidad\ memoria = 10 \times 21{,}1$$

$$Capacidad\ memoria = 211\ Mbits$$

$$Capacidad\ memoria = \frac{211\ Mbits}{8} = 26{,}375\ MB$$

Entonces se concluye que la capacidad necesitada no afecta en coste al modelo. Además tenemos alternativas hoy en día como almacenamiento locales muy pero muy grandes, orden de los TB, y almacenamientos en la nube sin coste por el orden de los GB. Si comparamos con el almacenamiento necesitado en el modelo base (28,625 MB), observamos que ha disminuido a 26,375 MB; en conclusión se mantiene dentro de los costes razonables a pagar para recibir los beneficios proporcionados por el mecanismo de control de interrupciones.

4.3. Análisis de la implantación experimental

Se utilizó la arquitectura propuesta en el capítulo 3, donde se desarrolló un sistema basado en agentes de software. En dispositivos inalámbricos se utilizó JADE-LEAP, que permitió el despliegue de este agente se lo realiza en modo partido, quiere decir que el contenedor en el que se ejecuta el agente se divide en una parte ubicada en el dispositivo móvil (Front-End) y otra parte ubicado en el servidor (Back-End), los cuales siempre están conectados.

JADE-LEAP utiliza el servicio de MAS (Sistema de Gestión Multiagente) que permite el diseño de la mensajería y RMI (invocación de métodos remotos). JADE apoya al manejo de interrupciones intermitentes mediante su protocolo MTP (Message Transport Protocol) que lleva a cabo el transporte físico de mensajes entre dos agentes que residen en contenedores distribuidos, brindando servicios de recuperación de fallos.

Al darse un quiebre en la comunicación, MTP realiza un monitoreo constante y cuando la reconexión se da a lugar, comunica a los agentes para que reinicien el dialogo desde el punto en que se generó la interrupción.

Ahora se pretende entregar a los usuarios incrementales para proveer al dispositivo móvil de una tecnología que permite la superposición, en tiempo real, de imágenes generadas por ordenador sobre imágenes del mundo real.

Estos datos superpuestos pueden ser tanto información relativa a elementos reales que están siendo visualizados, como datos independientes asociados a referentes físicos.

Nuestra intención es utilizar la funcionalidad de entregar datos superpuestos que contenga información del nivel de RSSI en ese punto geográfico que se encuentra el dispositivo móvil.

Con esto se quiere que el usuario actúe proactivamente y utilice como alternativa de consulta mapas online generados por otros usuarios, que permiten visualizar información de donde tenemos zonas de cobertura.

4.3.1. Proyección de patrones de software en los diagramas de diseño

En el diseño se define la arquitectura de la propuesta que soporte los requisitos. A partir del análisis se puede establecer las clases que forman parte del diagrama de clases del sistema (Figura 4.8-4.11). Presentamos los diagramas de clase del lado del cliente, que lo llamaremos *Agente Proxy Cliente* (*APC*) y son: aplicación, configuración, proxy y proxy-comportamiento. Los diagramas de clase del lado de servidor, que lo llamaremos *Agente Proxy Servidor* (APS) y son: proxy y proxy-comportamiento.

A esta clases se adicionó las nuevas clases que permiten la realizar la entrega de la información procesada por la regla de negocio Drools para que el usuario lo pueda visualizar con la tecnología de RA Look! y se genere el archivo necesario de mapa online con KML para su despliegue en Google Earth y Google Maps.

A continuación se explica la relación entre el modelo de patrones software y las clases que se han sido definidas en el diagrama de clases:

- Modelo y patrón *Observer*: Las clases que permiten el almacenamiento temporal y el observador que analiza constantemente el estado del buffer son: *BuscarVideo*, *SharedPreferences*, *InformacionWiFi*, *InformacionApp* y *KML*.
- Vista y patrón *Composite*: Las clases que son ejecutadas para lograr el despliegue del video streaming en el dispositivo cliente son: *VideoView*, *VideoPlayer*, *Uri*, *MediaController*, *LookAr*.
- Controlador: En este se define el patrón *Proxy* el cual está representado por las clases *JadeAplicacion*, *JadeGateway*, *AgenteProxy*, *GatewayAgent*, *Drools*, *ReglaNegocio*. El patrón *Strategy* también fortalece la capa de control implementado mediante las clases: *ThreadedBehaviourFactory*, *Behaviour* y *CyclicBehaviour*.

El observador externo que permite evaluar el estado de los recursos y del canal de comunicación está representado por las clases: *Socket*, *ConnectionListener*, *ListenerPuertoUDP*, *ListenerPuertoRTSP*, *ListenerMensajeUDP*, *ListenerMensajeRTSP*, *MessageTemplate*, *ACLMessage*, *LocationListener* y *SenalWiFi*.

Figura 4.8: Diagrama de clase APC con RA

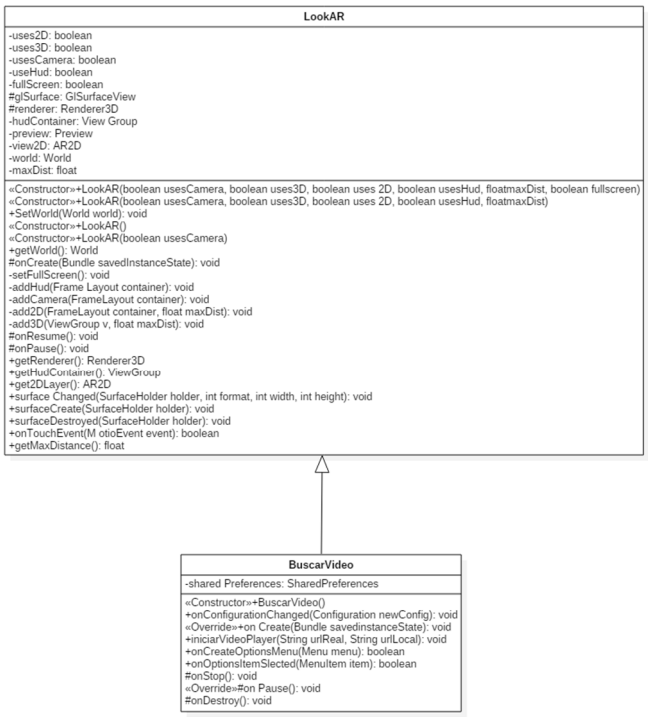


Figura 4.9: Diagrama de clase APC con RA

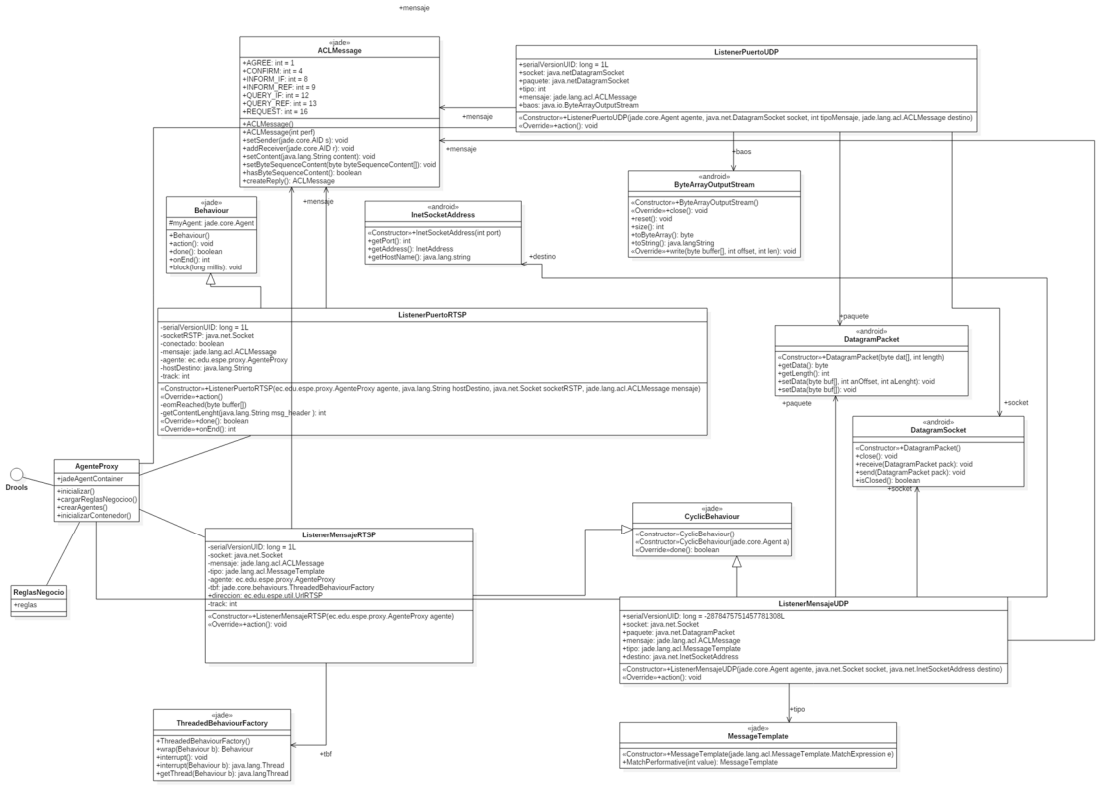


Figura 4.10: Diagrama de clases APS Aplicación con regla de negocio

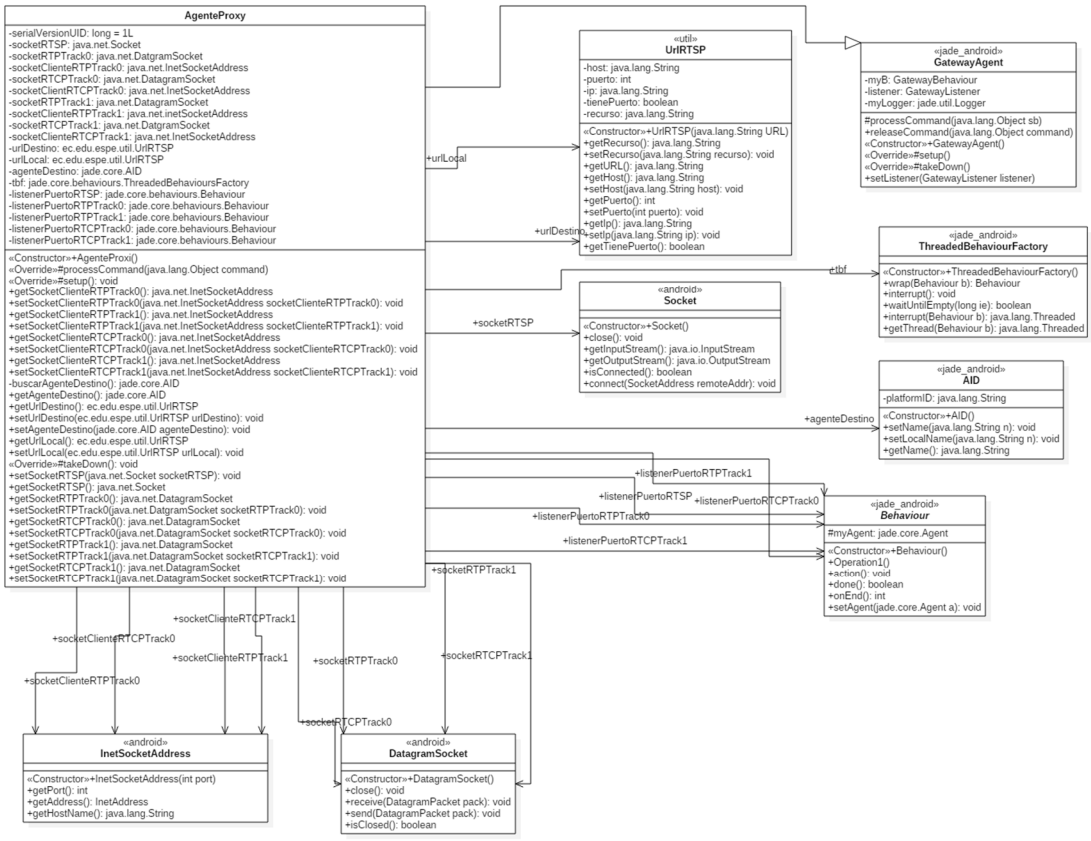


Figura 4.11: Diagrama de clase APS con regla de negocio

En la figura 4.12 se puede observar el diagrama de secuencia. Se explica a detalle la comunicación que se da entre las clases para cumplir el establecimiento de la sesión de video streaming, la activación de los observadores: del almacenamiento, de los recursos del dispositivo y del canal de comunicación, la definición de una interrupción y la selección del algoritmo que se ejecutara de acuerdo al evento ocurrido.

Además el tratamiento de la información procesada con reglas de negocio y entregada al cliente mediante técnicas de RA en la pantalla y la construcción del mapa online para despliegue en la nube mediante Google Earth y Google Maps.

Finalmente tenemos el diagrama de despliegue de la solución para control de interrupciones con RA y contiene dos componentes que son el servidor de aplicaciones y dispositivo inalámbrico.

Cada uno de ellos tiene distribuida las clases que conforman la solución y están distribuidas como se puede observar en la Figura 4.13.

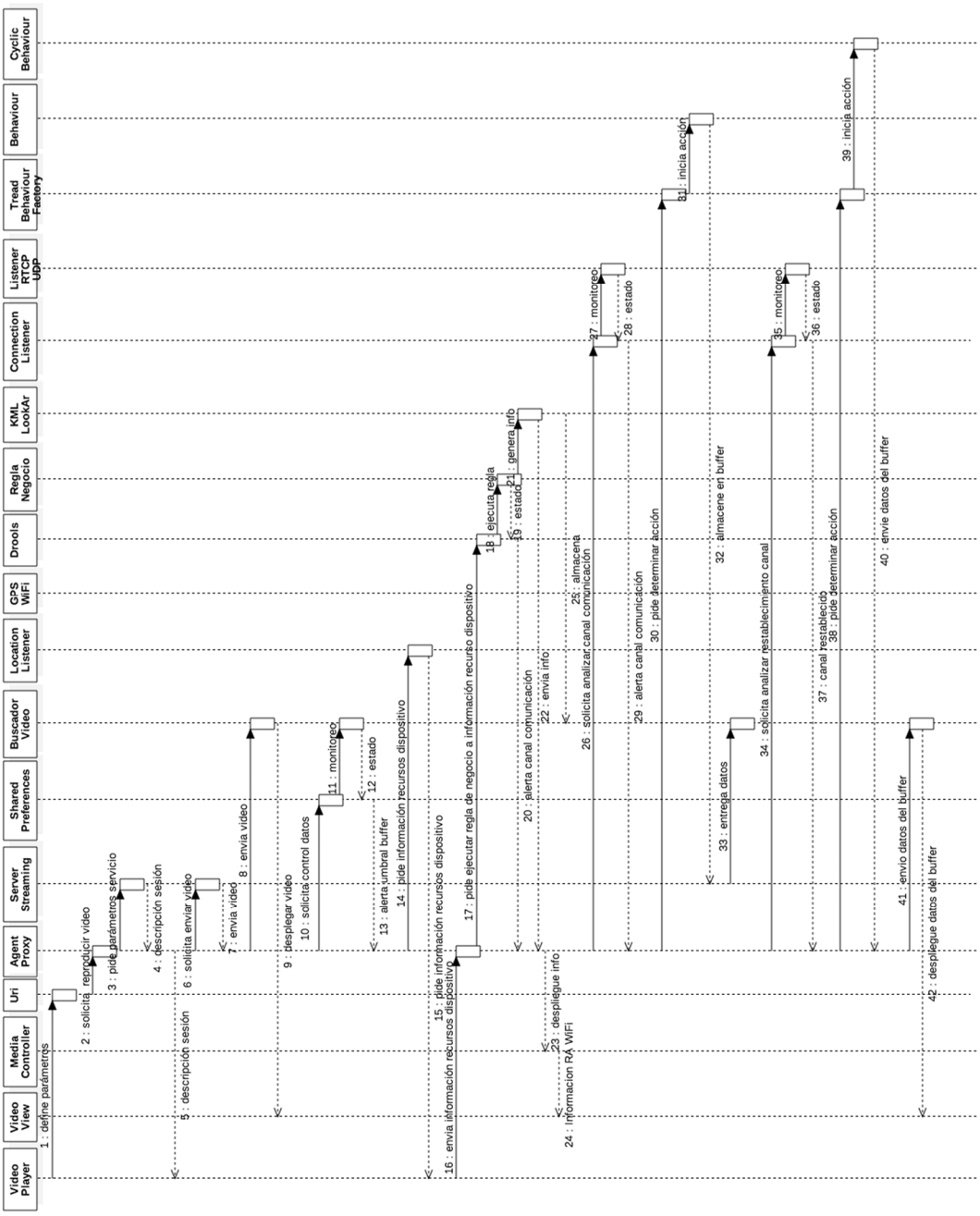


Figura 4.12: Diagrama de secuencia

El despliegue de información WiFi se la hace utilizando técnicas de RA mediante *LookAr* e *InformacionRA* en el *VideoView*. La creación del mapa online se lo hace mediante KML e *InformacionWiFi*. Dada la alerta de umbral *Proxy* solicita el servicio de *ConnectionListener* de observar el estado del canal de comunicación, si esta respuesta asegura que hay una interrupción el *Proxy* invoca al servicio del *ThreadBehaviourFactory* para que seleccione el algoritmo a ejecutar, en este caso *Behaviour*.

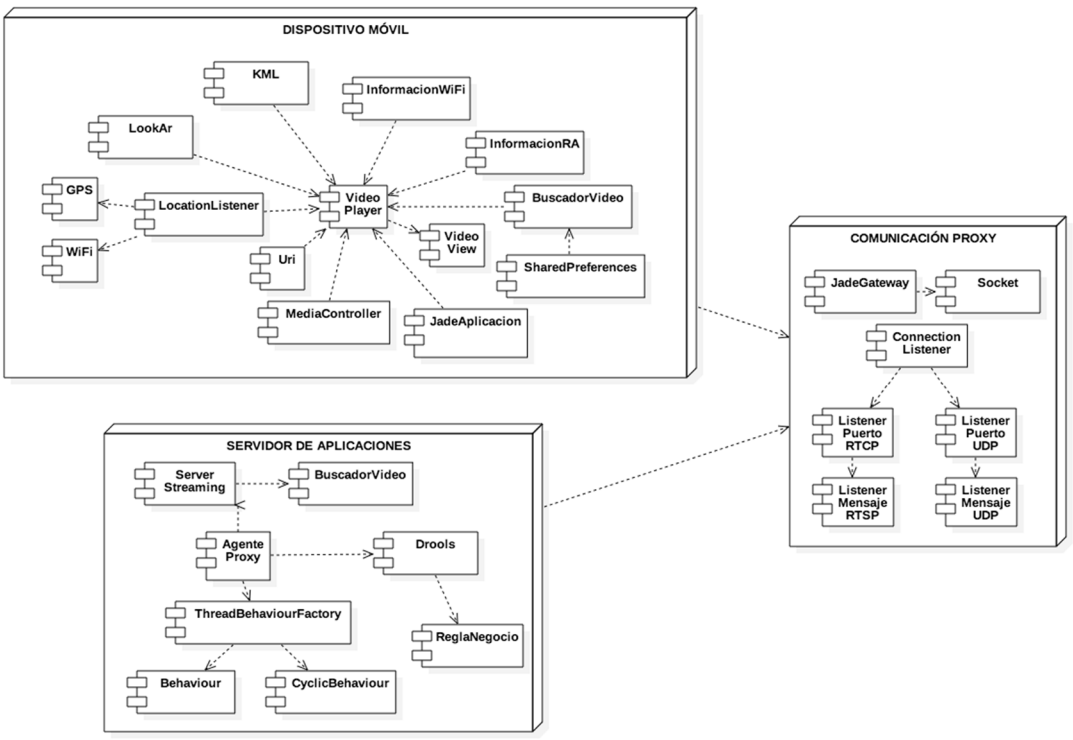


Figura 4.13: Diagrama de despliegue

Este algoritmo se encarga de que el *Proxy* solicite al *ServerStreaming* el almacenamiento temporal de los flujos que no pueden ser transmitidos y al *ConnectionListener* pide que evalúe el canal y determine su restablecimiento. Al restablecerse el canal el *Proxy* recibe esta notificación y pide al *ThreadBehaviourFactory* la ejecución del algoritmo correspondiente, para esta situación será *CyclicBehaviour* mediante el cual el *Proxy* solicita la transmisión del flujo que se encuentra almacenado en el Buffer del Servidor de Aplicación y sea entregado al proceso de visualización ubicado en el dispositivo móvil.

4.3.2. Desarrollo del modelo con software libre

El servidor de video streaming VLC se conecta con el APS y mediante acceso inalámbrico con el APC. En las comunicaciones establecidas entre los pares servidor–APS y APC–Cliente se transmiten mensajes utilizando el protocolo RTSP y para transmitir video el protocolo RTP. La comunicación entre el par APS-APC, que se despliegan en la

plataforma JADE, se comunican a través de mensajes FIPA ACL, simplificando la comunicación entre agentes.

El APC se divide en dos partes: el Front-End que actúa con el dispositivo móvil Android y el Back-End con el servidor de video streaming. La comunicación entre estas partes es mediante un canal de comunicación inalámbrica, donde las posibles interrupciones se gestionan de forma transparente por el APC.

Además se implementa de forma transparente dos funciones: el mecanismo de almacenamiento y envío, y el intercambio de mensajes de negociación RTSP y los paquetes RTP y RTCP. Con este se garantiza que el agente de dispositivo móvil nunca se va a desconectar del cliente.

El APS recepta los mensajes del APC y los envía al servidor según los puertos establecidos en la negociación y recepta los mensajes provenientes del servidor para renviarlos al APC

El uso de señalización MTP entre los agentes APS-APC permite definir si el dispositivo móvil está fuera de cobertura en una comunicación inalámbrica y reanudar automáticamente la sesión de video streaming.

Al ocurrir una interrupción, los mensajes FIPA-ACL ya no se transmiten al dispositivo móvil y son almacenados en el buffer del APS, para ser entregados al APC una vez restablecida la conexión.

Estos mensajes almacenados en el APS son encapsulados y tratados por el servicio de reparto persistente hasta que el agente APC se conecte y puedan ser enviados. Con este proceso se hace transparente el comportamiento de interrupción de la red inalámbrica al servidor y al cliente.

El protocolo MTP permite definir el tiempo de espera para intentar reconectar APS-APC, cuyo valor que por defecto es un minuto. Se debe considerar éste tiempo para el dimensionamiento del buffer.

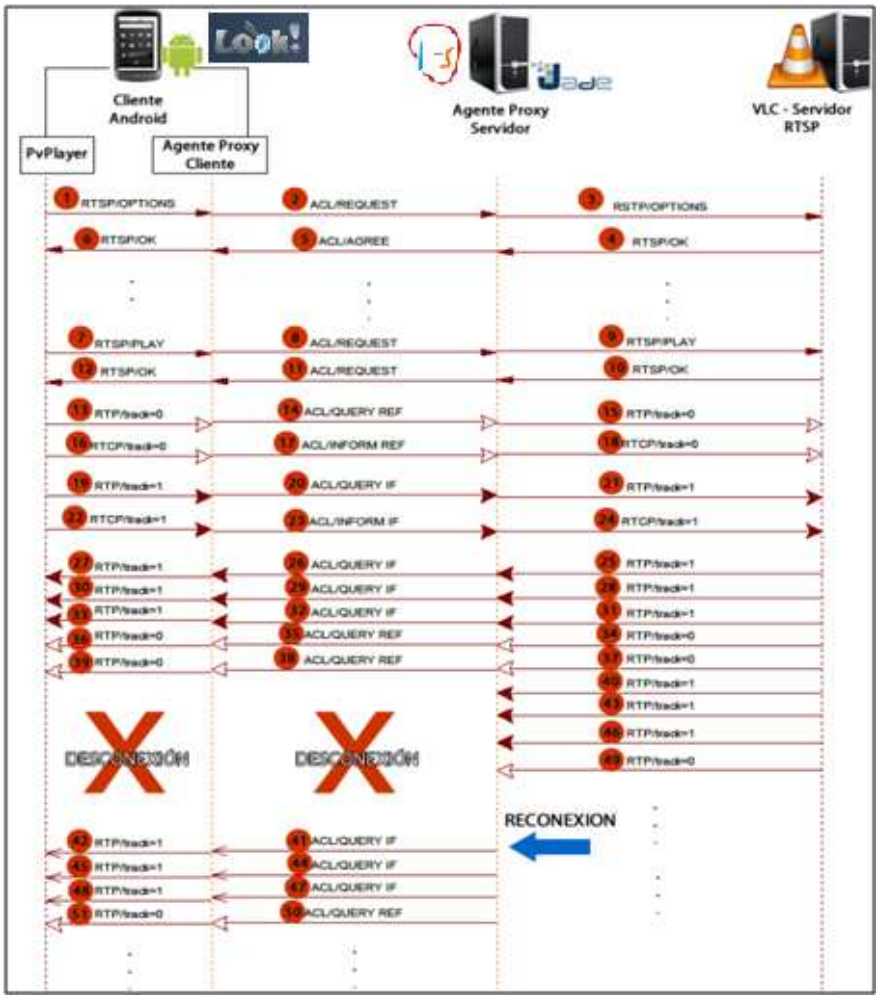


Figura 4.14: Arquitectura JADE con RA y reglas de negocio

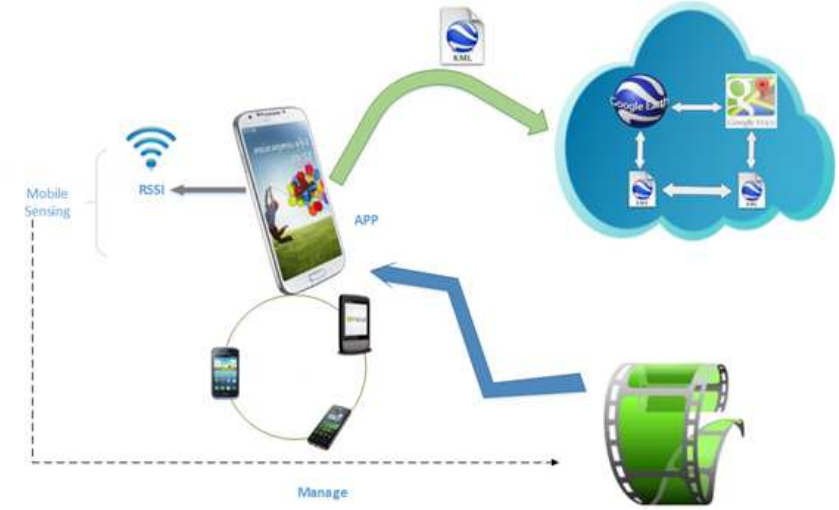


Figura 4.15: Arquitectura con RA y reglas de negocio

La Figura 4.15 indica cómo está constituida la nueva arquitectura. Empezamos integrando al agente APS el servidor las reglas necesarias para que, al momento de formar la cadena de conexión entre el servidor video streaming y el APC, la aplicación provea de los datos censados por los sensores de geo localización y RSSI, y este pueda tomar la acción pertinente definida en la regla de negocio.

En el APC se implementó la clase *LocationListener* que monitorea los cambios de RSSI mediante *RSSI_Changed_Action*() y lo entrega a APC para éste a su vez lo envíe mediante mensajes FIPA-ACL a APS. Es importante indicar que por cada cambio de RSSI asocia su ubicación geográfica mediante el uso de la clase *LocationListener*, quien primero verifica que se encuentre activado el GPS y solicita a *onLocationChanged*() la actualización de los métodos *getLatitude*(), *getLongitude*(), *getAltitude*() y *getAccuracy*(), para obtener la latitud, longitud, altitud y precisión respectivamente. En APS la información se almacena para ser tratada con la respectiva regla de negocio dado un cambio en el valor de RSSI.

En el APS recibida la información de los estados de RSSI y GPS, valida si ha existido algún cambio de RSSI con respecto al anterior almacenado. Si existe cambio solicita se ejecute la regla de negocio, mediante la clase *Drools ReglaNegocio*; a los valores de RSSI, los procesa y entrega al APC para despliegue de la información en pantalla mediante RA.

Se ha utilizado como valores de referencia del RSSI para la aplicación de la regla de negocio la siguiente información:

- Valores por encima de un -70dBm = No vamos a conectar y si lo hacemos será con caídas constantes.
- Valores por encima de un -50dBm = Caídas aleatorias.
- Valores por encima de un -40dBm = Conexión aceptable pero podemos esperar perdida de paquetes.
- Valores por encima de un -30dBm = Conexión buena.
- Valores por debajo de -27dBm = Conexión excelente máxima velocidad de transferencia.

Con esta consideración hemos definido lo siguiente para la aplicación de la regla de negocio:

- Valores por encima de un -40dBm: presentar en pantalla mensaje de *Entrando en zona de interrupción* y almacenar video streaming en buffer del APS, asociar al punto geográfico el color rojo para el dibujo de la ruta.
- Valores entre -30dBm y -40dBm: presentar en pantalla mensaje de *Verifique nivel de RSSI*, asociar al punto geográfico el color naranja para el dibujo de la ruta.
- Valores por debajo de un -30dBm: presentar en pantalla mensaje *Video streaming recibido*, asociar al punto geográfico el color verde para el dibujo de la ruta.

La RA dentro de la aplicación permitió generar y hacer transparente al usuario el tratamiento de datos, mediante el servicio de persistencia de datos que mediante el uso del almacenamiento local, almacena los datos de la aplicación en el dispositivo durante distintas ejecuciones y también permite el tratamiento de los datos compartidos entre distintas aplicaciones. Se la implementó utilizando el framework Look! y utilizando la clase *LooKAr* se construyó los elementos a ser presentados en la pantalla del usuario.

Finalmente con la información procesada por la regla de negocio y entregada esta información a APS, éste solicita convertir la información para generar archivos KML que contienen información de geo localización, RSSI e imagen del sector donde está el móvil, para subirlo a la nube y proveer el ambiente colaborativo. Esta información puede ser consultada por los usuarios del campus universitario ESPE a través de aplicación Google Earth (Figura 4.16). Entonces el *PLAYER* del sistema operativo Android se modificó para que integre un botón que permita al dispositivo móvil cambiar al diagrama de cobertura cooperativo que han realizado entre todos los usuarios en el campus universitario ESPE, con lo cual si ubicas la posición del móvil el usuario puede saber porque no está recibiendo el video streaming (porque posiblemente este en una zona de interrupción) y además pueda moverse de tal manera que pueda ingresar a una zona con cobertura (Figura 4.17-4.18).

Figura 4.16: Aplicación Google Earth con despliegue KML

Figura 4.17: Aplicación Google Earth con despliegue de KML

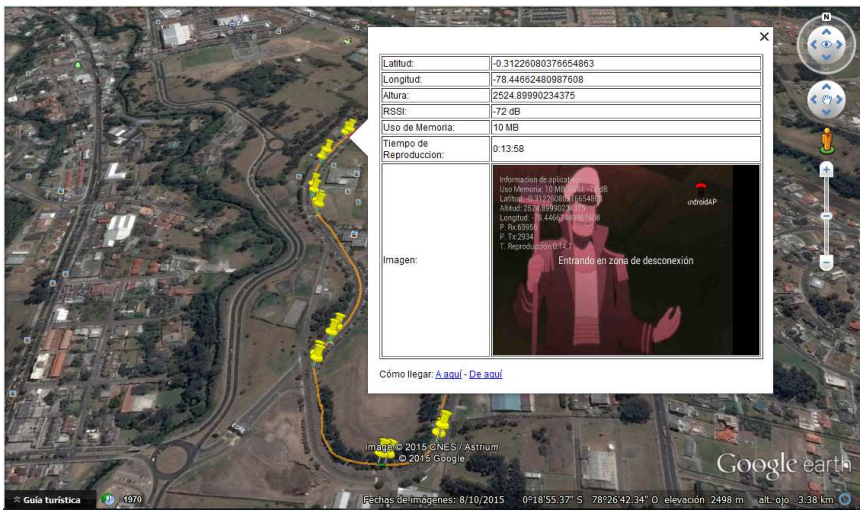


Figura 4.18: Aplicación Google Earth con despliegue de KML

4.3.3. Resultados experimentales

Para realizar las pruebas se ha utilizado los siguientes elementos:

- Una red inalámbrica con tecnología WiFi con un punto de acceso de 54Mbps.
- Un equipo portátil, procesador AMD de 2.20GHz, 4 GB de RAM y tarjeta inalámbrica Atheros AR5009 IEEE 802.11 a/g/n WiFi Adapter; en el cual se ejecuta el Agente Proxy Servidor APS.
- Un dispositivo móvil Samsung Galaxy modelo S2, Procesador Dual Core Cortex A9 1.2GHz, 1GB RAM, Tarjeta microSD de 16GB expandible hasta 32GB, IEEE 802.11b/g/n; en el cual se ejecuta el Agente Proxy Cliente APC.

Antes de realizar las pruebas, verificamos el paso mensajes ACL entre los agentes proxy, con el sniffer propio de la plataforma JADE, teniendo un resultado exitoso, ya que los mensajes se pasan de manera correcta, sin errores, como se puede ver en la Figura 4.19.

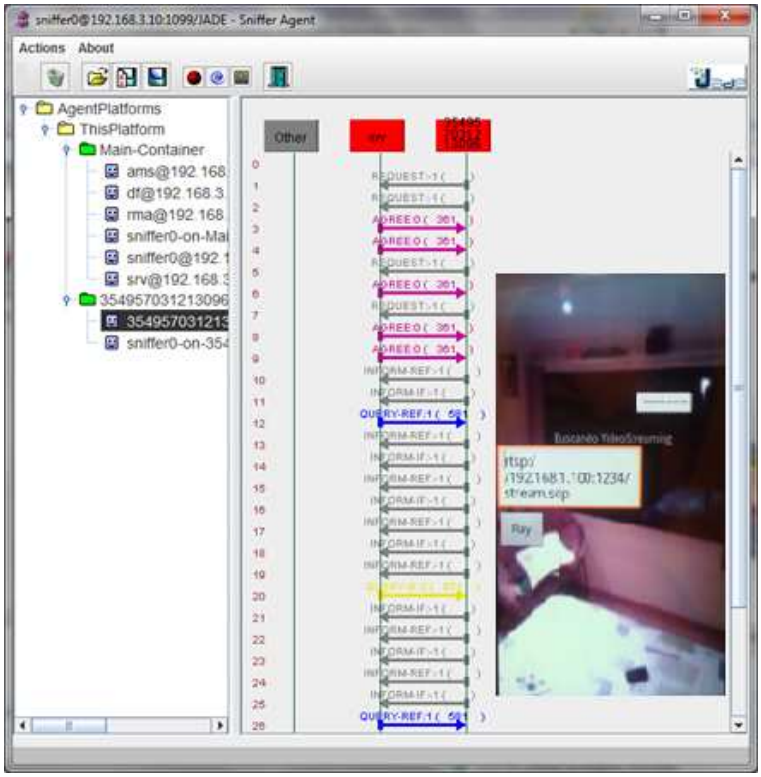


Figura 4.19: Paso de mensajes FIPA-ACL

Además se instaló una herramienta sniffer WireShark, que es la que facilita obtener los paquetes de audio y video con toda su información como tiempo, bytes, timestamp, entre otros. El sniffer está ubicado en el cliente móvil Samsung Galaxy S2. Para que esta aplicación funcione se debió dar permisos de root.

El servidor RTSP que se ha usado es el VLC y se ha decidido utilizar la configuración para emitir el audio y video indicada en las tablas 4.14 y 4.15.

Tabla 4.14: Configuración para emitir video

CODEC	Tasa de Bits	fps	Ancho	Alto
H264	600 kb/s	25	480	320

Tabla 4.15: Configuración para emitir audio

CODEC	Tasa de Bits	Canales	Tasa de muestreo
AAC	128 kb/s	1	44100

El servidor VLC requiere para emitir video por video streaming debe usar la siguiente línea de configuración para emitir cualquier formato de video:

```
:sout=#transcode{soverlay, ab=64, samplerate=44100,channels=2, acodec=mp4a, vcodec=h264,
width=480,height=320,fps=25,vb=400,venc=x264{vbv-bufsize=10000, partitions=all,level=12,no-
cabac, subme=7, threads=4, ref=2,mixed-refs=1,bframes=0,min-keyint=1,keyint=50,trellis=2,
direct=auto, qcomp=0.0, qpmax=51}}:gather:rtp{mp4a-latm, sdp=rtsp://0.0.0.0:1234/stream.sdp}
```

Se recomienda tener un video previamente codificado en el formato adecuado para evitar la sobrecarga del servidor, emitiéndolo con la siguiente línea:

```
:sout=#rtp{sdp=rtsp://0.0.0.0:1234/stream.sdp} :sout-keep
```

Debe ejecutarse el servidor JADE para posteriormente instalar la App en el dispositivo móvil, asegurando la conexión entre servidor y cliente en la red. Dentro de la interfaz gráfica de la aplicación se visualiza un cuadro de texto, en el cual se debe ingresar la dirección del servidor video streaming. En opciones activar la localización GPS, activar proxy e ingresar la dirección IP y puerto del servidor proxy (JADE) y finalizar pulsando el botón reproducir.

Para poder observar las ventajas o desventajas del software desarrollado se ha tomado dos escenarios:

- Escenario 1: con proxy (Jade Video streaming con RA).
- Escenario 2: sin proxy.

En ambos escenarios se ha emitido un clip de video (.avi), con una duración de 02:27 minutos, así como también se calculará una muestra de los paquetes de audio y otra de video, para cada software. Para calcular la muestra en los dos escenarios el único valor que cambia es N, que es el tamaño total de paquetes recibidos. Los resultados se muestran en las tablas .4.16 y 4.17.

Fórmula a utilizar:

$$n = \frac{Z^2 p q N}{N E^2 + Z^2 p q}$$

Donde:

Z = 0,98

p = 0,50

q = 0,5

E = 0,02

Tabla 4.16: Escenario con proxy

	N (# total de Paquetes Recibidos)	**n(muestra)**
Audio	2246	481
Video	3457	511

Tabla 4.17: Escenario sin proxy

	N (# total de Paquetes Recibidos)	**n(muestra)**
Audio	2446	481
Video	3457	511

Se realizó la experimentación y toma de la muestra para los dos escenarios indicados en las áreas exteriores de la ESPE, para valorar los parámetros de jitter y retraso.

Jitter

La Figura 4.20 muestra el jitter de paquetes de audio.

La Figura 4.21 muestra el jitter de paquetes de video.

Retraso

La Figura 4.22 muestra el retraso de paquetes de audio.

La Figura 4.23 muestra el retraso de paquetes de video.

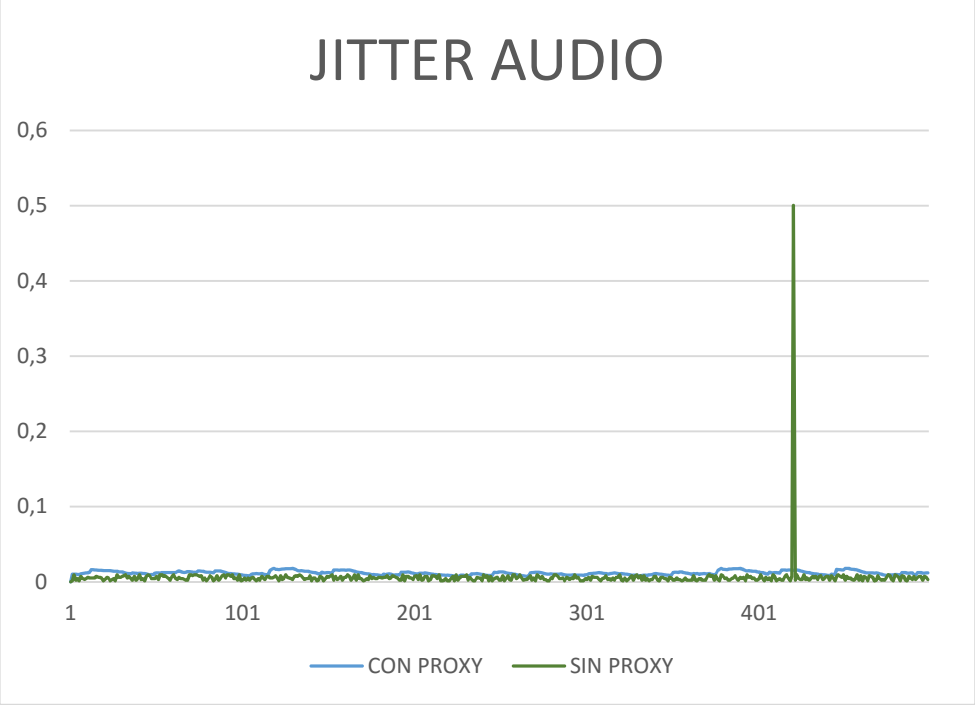


Figura 4.20: Jitter de paquetes de audio

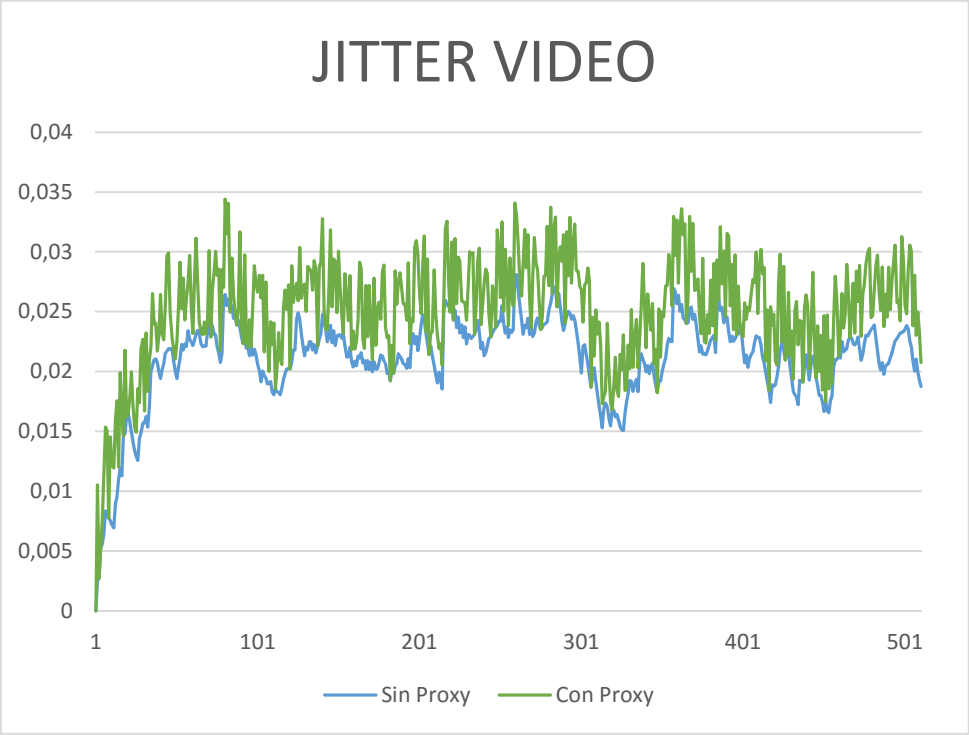


Figura 4.21: Jitter de paquetes de video

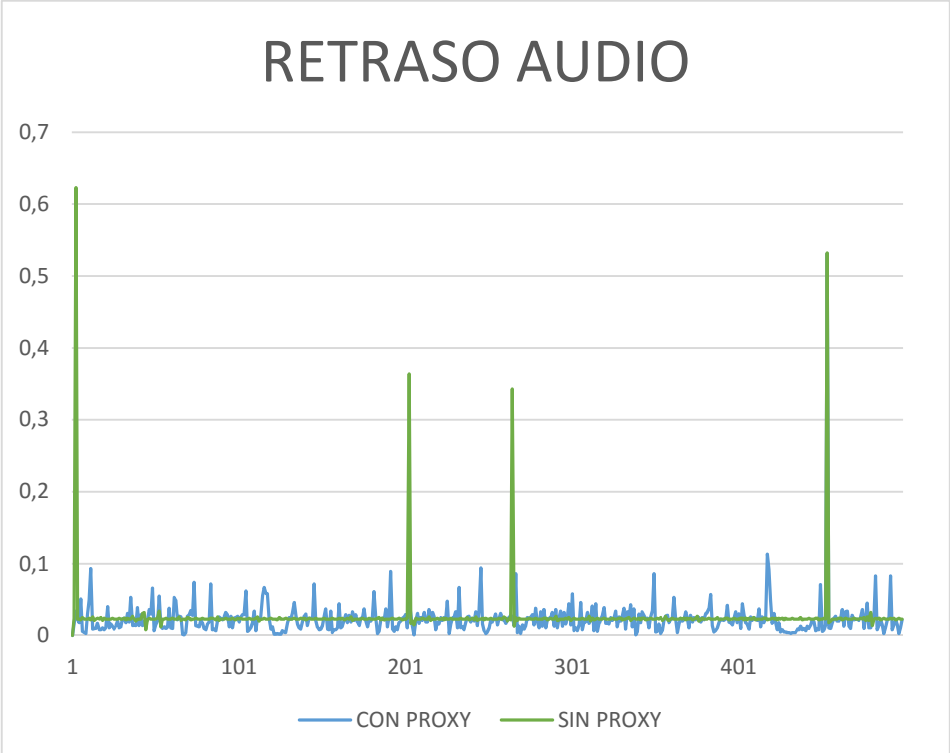


Figura 4.22: Retraso paquetes de audio

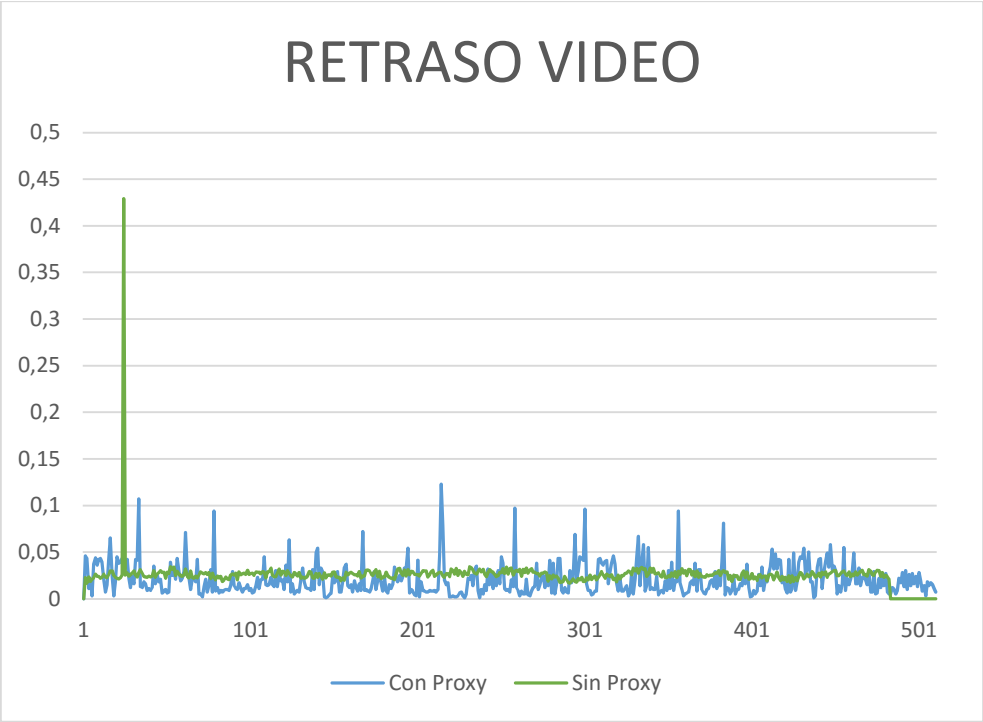


Figura 4.23: Retraso de paquetes de video

Tanto los gráficos del jitter y del retraso nos demuestran que la aplicación lleva paquetes más grandes, pues poseen mayor información como es la de control, pero si se puede observar mayor estabilidad en la comunicación, ya que no existe interrupciones.

Los resultados obtenidos es la reproducción de audio y video, con calidad al 100%. Al momento de salir del área de cobertura por aproximadamente 30 segundos, los paquetes tanto de audio y video se recuperan de manera exitosa, sin pérdida de ningún paquete, pero si con una demora al momento de reconectarse, debido a que los paquetes que se han quedado guardados en el buffer deben ser despachados y recalculados su retraso.

Ahora adicionamos dos casos más de experimentación realizados en mi residencia en Sangolquí, Ecuador y en la Universidad de las Palmas de Gran Canaria.

Se realizó un experimento en los exteriores de mi residencia; se puede apreciar como el RSSI presenta sus variaciones conforme se acerca o aleja de los puntos de

acceso inalámbrico que dispone el conjunto habitacional donde resido. Las Figura 4.24 y 4.25 presentas estos resultados.

Finalmente se realizó una última prueba en la Universidad de las Palmas de Gran Canaria, realizando una trayectoria en los exteriores entre el Edificio del Departamento de Telemática y el Edificio de Arquitectura, se puede apreciar el comportamiento de RSSI en toda la trayectoria seguida y los resultados se muestra en la Figura 4.26.

Figura 4.24: Prueba realizada en residencia

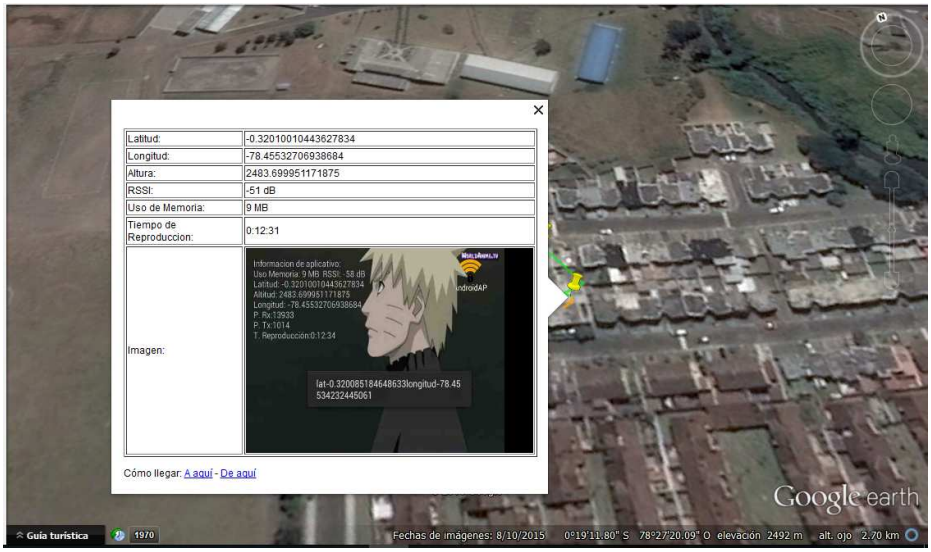


Figura 4.25: Prueba realizada en residencia

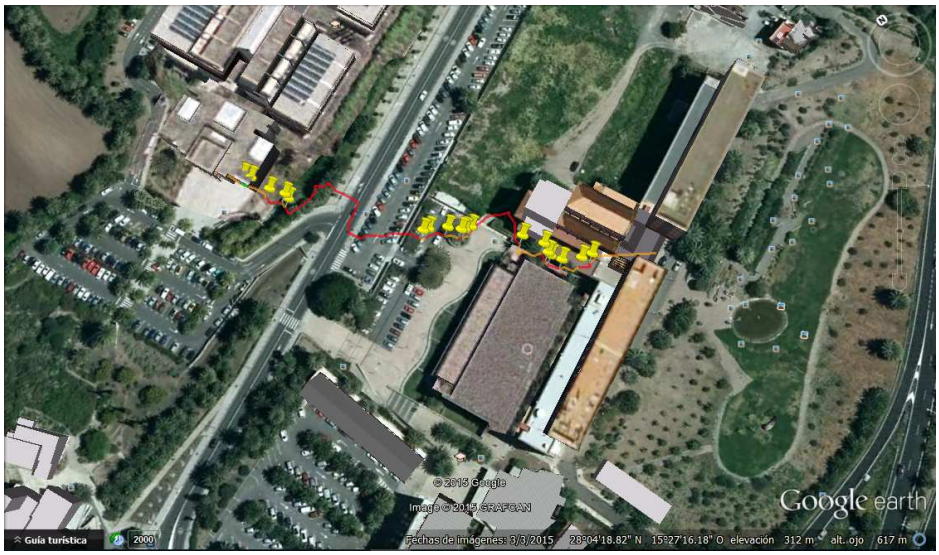


Figura 4.26: Prueba realizada en la ULPGC

La solución desarrollada presenta un comportamiento inteligente para predecir una interrupción inalámbrica y permitirle al usuario actuar proactivamente para que pueda retornar a una zona de cobertura. Si la interrupción se da, la solución permite guardar el flujo de video que no puede ser entregado al usuario. Este es entregado una vez que se reanude la comunicación, manteniendo la continuidad del servicio.

Se ha probado que la solución permite controlar las interrupciones del servicio de video streaming en ambientes inalámbricos, disminuyendo el impacto negativo que ocasiona una interrupción del servicio en el usuario; ya que reduce la pérdida de tiempo que tendría el usuario en la reproducción del video con interrupciones y evita el abandono de la sesión.

5. Solución para video en tiempo real con control de interrupciones

En este capítulo se describe la funcionalidad incrementada al modelo base con el objetivo de lograr su despliegue en cualquier tipo de dispositivo móvil (teléfono inteligente o computador) para reproducir video en tiempo real y lograr el control de interrupciones.

5.1. Análisis del diseño basado en patrones software

El sector de la videoconferencia está viviendo una auténtica revolución desde hace un par de años. La aparición de servicios de videoconferencia en la nube, está desbancado totalmente las soluciones tradicionales que requerían de la compra de equipos muy costosos y que a menudo se quedaban obsoletos antes de amortizarse por completo. La videoconferencia tradicional conecta un punto con otro, es decir, una sala de videoconferencia y su equipo correspondiente, con otra sala de las mismas características, restringiendo el uso a terceros (clientes, proveedores…), siendo el acceso desde el exterior limitado en muchas ocasiones por incompatibilidad de redes y sistemas. Sin embargo, la videoconferencia en la nube permite conectar diferentes participantes desde sus ordenadores, dispositivos inalámbricos o equipos de sala tradicionales, sin importar el lugar donde estén, simplemente se necesita una conexión a Internet de banda ancha, ganando en flexibilidad, versatilidad y con total seguridad para la red de la empresa al residir el servicio en la nube.

La videoconferencia en la nube es una herramienta de comunicación esencial y estratégica para todo tipo de empresas y organizaciones, ya que proporciona importantes beneficios a nivel de reducción de costes y aumento de la productividad de los usuarios. Existen sistemas que proveen de poderosísimas herramientas que habilitan el uso de aplicaciones desde un browser a otro para llamadas de voz, chat de video, o lo que nosotros conocemos como videoconferencia- e incluso un compartidor de archivos en formato P2P, todo ello sin la necesidad de descargar ningún tipo de plugin.

En estos sistemas durante el proceso de transmisión del video en tiempo real, podrían existir rupturas o cortes especialmente en comunicación con redes inalámbricas; sin embargo, si se le provee de un mecanismo de control de interrupciones se podría privilegiar la usabilidad del sistema, es decir, sería deseable que al sufrir una interrupción el propio aplicativo identifique el restablecimiento de la comunicación y continúe el proceso de videoconferencia sin que el usuario vuelva a reiniciar la sesión y entregando el video almacenado que no pudo visualizar.

Se ha probado que una interrupción demanda tiempos adicionales de procesamiento y una alta cantidad de flujos que se retransmiten, lo cual ha sido

reducido con la implementación de agentes de software que notifican la posibilidad de una interrupción y permiten definir acciones proactivas para enfrentar este problema, logrando mejorar la QoE del usuario.

El modelo base del capítulo 3 ha podido ser implementado en plataforma específica, Android. Estamos interesados en mantener la arquitectura de software del modelo base para control de interrupciones, que permita implantar un mecanismo de control de interrupciones en los sistemas de transmisión de video en tiempo real sobre. El mecanismo desarrollado en el capítulo 3 permite controlar las interrupciones en ambientes Android. Pero depende de descargar el plugin, instalarlo en el dispositivo móvil Android para disponer de este mecanismo.

Existen nuevos sistemas para videoconferencia en la nube que carecen de un control de interrupciones. Si se le provee de un mecanismo de control de interrupciones se podría privilegiar la usabilidad del sistema, es decir, sería deseable que al sufrir una interrupción el propio aplicativo identifique el restablecimiento de la comunicación y continúe el proceso de videoconferencia sin que el usuario vuelva a reiniciar la sesión y entregando el video almacenado que no pudo visualizar.

En la Figura 5.1 se observa los requisitos funcionales que deben incluirse en el diseño final, los cuales se han establecido mediante diagrama de casos de uso y su descripción específica. El primer caso de uso describe el inicio de la petición de videoconferencia, explicada en la Tabla 5.1. El segundo caso de uso describe la solicitud de monitoreo del umbral de almacenamiento temporal, explicada en la Tabla 5.2. El tercer caso de uso describe la petición de evaluar el comportamiento del canal de comunicación, para confirmar si se trata de una interrupción, explicada en la Tabla 5.3. El cuarto caso de uso describe la necesidad de mantener y gestionar la continuidad del servicio de videoconferencia por en evento del canal de comunicación, explicada en la Tabla 5.4. El quinto caso de uso describe las actividades a cumplirse para gestionar una interrupción del servicio de videoconferencia, explicada en la Tabla 5.5. El sexto caso de uso describe la actividades a cumplirse cuando el canal de comunicación se ha restablecido y se de inicio a la reconexión de la sesión de videoconferencia, explicada en la Tabla 5.6.

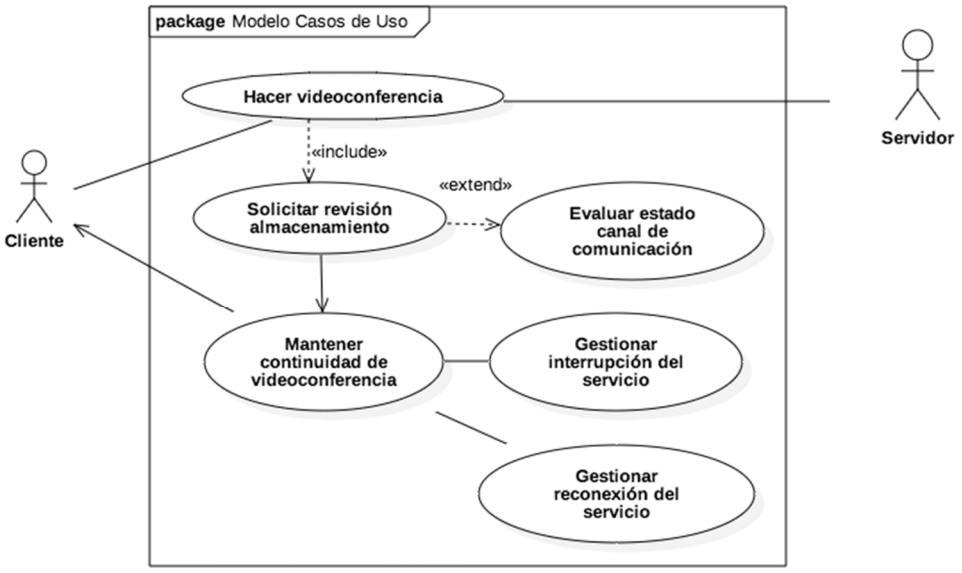


Figura 5.1: Casos de uso modelo para video en tiempo real

Tabla 5.1: Casos de uso modelo – hacer videoconferencia

Caso de Uso: hacer videoconferencia
Actores: cliente Web, servidor
Resumen: describe inicio petición de videoconferencia.
Precondiciones:
Descripción: 7. El cliente Web solicita servicio de videoconferencia. 8. Ingresa datos IP del servidor. 9. Los datos son validados por el proxy. 10. El servidor pide acceso a recursos dispositivo (cámara y micrófono). 11. El cliente Web autoriza acceso a recursos. 12. Si no existen problemas el proxy solicita el servicio al servidor. 13. El cliente solicita registro de usuario. 14. El proxy valida usuario y registra. 15. Si usuario existe, solicita a cliente que indique si quiere reutilizar usuario. 16. El cliente solicita establecer videoconferencia con otro usuario. 17. El proxy pide ID de usuario al servidor 18. El servidor asocia ID a usuario y lo registra. 19. El proxy recibe información de registro. 20. El proxy solicita al servidor establecer sesión de videoconferencia. 21. El servidor indica sesión establecida 22. Se despliega una ventana en cada cliente para inicio videoconferencia. 23. Inicia entrega de flujos de datos entre clientes Web..
Pos condiciones: solicita la revisión constante del almacenamiento.
Observaciones:

Tabla 5.2: Casos de uso modelo – revisión almacenamiento

Caso de Uso: solicitar revisión almacenamiento
Actores:
Resumen: explica la importancia de monitorear umbral de almacenamiento temporal
Precondiciones: el proxy solicita la revisión del almacenamiento
Descripción: 9. Se pide constantemente información del umbral del almacenamiento. 10. Se compara con el máximo nivel establecido. 11. Se genera una alerta en caso de superación del umbral definido. 12. Se comunica este problema al proxy
Pos condiciones: solicita la evaluación del canal de comunicación
Observaciones:

Tabla 5.3: Casos de uso modelo – canal de comunicación

Caso de Uso: evaluar estado canal de comunicación
Actores:
Resumen: para confirmar que se trata de una interrupción, se evalúa el estado del canal de comunicación.
Precondiciones: proxy solicita evaluación del canal de comunicación
Descripción: 7. Se evalúa el canal de comunicación. 8. Se determina el estado del canal de comunicación 9. Este estado es comunicado al proxy.
Pos condiciones: se solicita mantener la continuidad del servicio de videoconferencia.
Observaciones:

Tabla 5.4: Casos de uso modelo – continuidad video

Caso de Uso: mantener continuidad de videoconferencia
Actores:
Resumen: dado un evento del canal de comunicación debe gestionarse la continuidad del servicio de videoconferencia.
Precondiciones: el proxy solicita ejecución de proceso de acuerdo al estado del canal de comunicación.
Descripción: Alerta interrupción : 5. En caso de ser una alerta de interrupción se solicita la gestionar interrupción del servicio. Alerta conexión: 6. Cuando el canal se ha restablecido se solicita gestionar reconexión.
Pos condiciones:
Observaciones:

Tabla 5.5: Casos de uso modelo – gestionar interrupción

Caso de Uso: gestionar interrupción del servicio
Actores:
Resumen: explica las actividades a cumplir cuando ocurre una interrupción y como se controla la misma.
Precondiciones:
Descripción: 7. Se solicita el almacenamiento de los flujos en el buffer del cliente. 8. Se despliega en la interfaz gráfica del usuario el mensaje "Reconectando" por el problema dado y que se encuentra en estado de espera para reconectarse automáticamente 9. Se pide el servicio de evaluar continuamente el canal de comunicación.
Pos condiciones:
Observaciones:

Tabla 5.6: Casos de uso modelo – gestionar reconexión

Caso de Uso: gestionar reconexión
Actores:
Resumen: explica las actividades a cumplir cuando el canal se ha restablecido.
Precondiciones:
Descripción: 5. El proxy solicita restablecer sesión al servidor. 6. El servidor inicia restablecimiento de sesión. 7. El proxy solicita que los flujos almacenados en el buffer del cliente sean transmitidos al servidor. 8. El servidor recibe el flujo de datos y pide realizar transcodificación. 9. El servidor observa los ID de usuario y envía la información a cada cliente Web. 10. El cliente Web automáticamente continúa visualizando la videoconferencia y se despliega una nueva ventana con el flujo de datos enviado por el servidor en el punto en el que se encontraba cuando ocurrió la interrupción.
Pos condiciones:
Observaciones:

5.1.1. Justificación de los patrones software a utilizar

Se planteó el modelo base basado en patrones software de diseño, en especial *Proxy*, que contribuyeron a establecer un mecanismo que permitió gestionar eficientemente el control de una interrupción por un quiebre del canal de comunicación. Este diseño permitió la organización estructurada del modelo por capas MVC y se potenció con el

uso de patrones software de diseño como *Observer*, *Strategy* y *Composite*. Nuestro interés es mantener el diseño anterior y definir componentes incrementales que posibiliten controlar las interrupciones en sistemas de video en tiempo real.

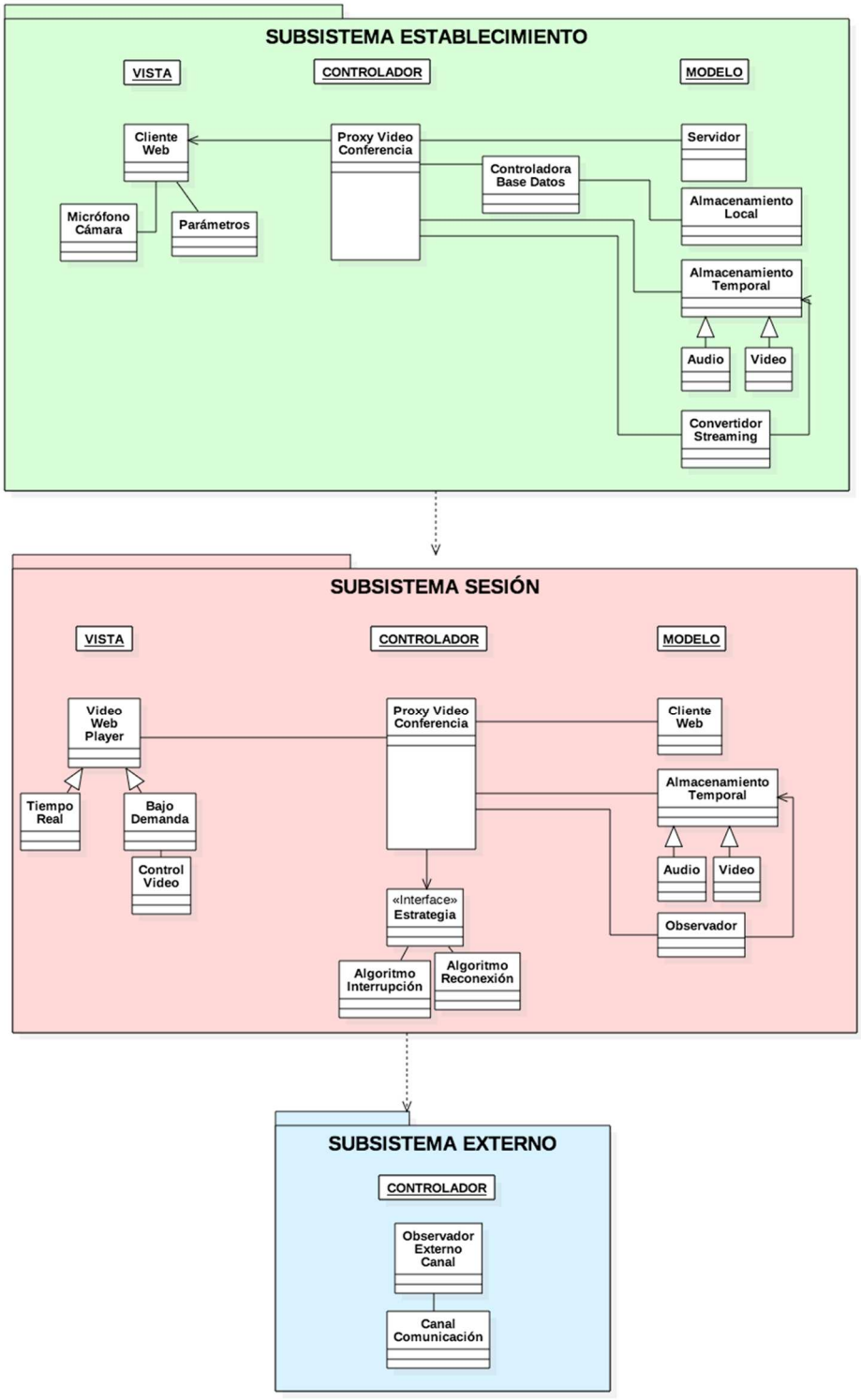


Figura 5.2: Modelo para video en tiempo real

En la Figura 5.2 se puede observar el incremental funcional en el subsistema establecimiento, que permiten establecer, administrar usuarios y garantizar la continuidad de la videoconferencia. Para lograr el establecimiento se requiere incorporar una controladora de base de datos que permite la administración de usuarios del sistema de video conferencia y se utiliza el patrón *Adapter* para transcodificar los flujos almacenados por los clientes en la interrupción y compartirlos con cada uno, garantizando la continuidad de la videoconferencia.

El modelo se encuentra definido por tres subsistemas: establecimiento; sesión y externo, y tres patrones software de diseño: MVC, *Observer* y *Adapter*. El MVC se mantuvo para lograr independizar las capas de desarrollo, mejorar la gestión de cambio y escalabilidad. El patrón de diseño *Observer* es el encargado de analizar el estado del canal de comunicación. El patrón de diseño *Adapter* es el encargado de convertir el audio y video almacenado temporalmente en el buffer del servidor Web, y una vez convertido enviarlo al almacenamiento temporalmente en cada cliente peer y su posterior despliegue.

El subsistema establecimiento, presenta los siguientes elementos en sus capas de:

- *Modelo*: se encuentra ubicado en el servidor Web, encargado de proveer almacenamiento temporal para recibir los flujos de datos almacenados por los clientes Web, en una interrupción y almacenamiento local de la base de datos de usuarios. Este modelo adiciona y se fortalece con la implementación del patrón de diseño adaptador, que convierte los datos de audio y video almacenados en flujos para ser enviados a los clientes Web.
- *Vista:* se despliega en los dispositivos inalámbricos utilizando el patrón de diseño *Composite*. Es una interfaz de usuario que presenta los formularios para: permitir el acceso a los recursos de la cámara y micrófono de cada dispositivo móvil; creación de usuario de videoconferencia e inicio de videoconferencia.
- *Controlador*: se ubica en el servidor Web, está conformado por el patrón de diseño *Proxy* y la controladora de base de datos. El *Proxy* se encarga de validar

la existencia de usuario mediante la controladora y de establecer la videoconferencia Web mediante el servidor.

El subsistema sesión, presenta los siguientes elementos en sus capas de:

- *Modelo*: se encuentra ubicado en cada cliente Web, encargado de proveer almacenamiento temporal para guardar los datos de audio y video capturados por la cámara y micrófono del cliente, si presenta una interrupción del canal. Este modelo se fortalece con la implementación de un patrón de diseño *Observer* que constantemente está monitoreando el estado del buffer. Cuando el buffer supera el umbral establecido, se encarga de notificar al *Proxy* de este evento.
- *Vista*: se despliega en los dispositivos inalámbricos utilizando el patrón de diseño *Composite*. Es una interfaz de usuario que presenta los formularios para finalizar video conferencia y salir plataforma; además del despliegue de la videoconferencia y de las estadísticas de red del servicio de videoconferencia. En caso de ocurrir una interrupción, despliega un mensaje *Reconectando*, el cual permanece activo hasta cuando el canal se restablezca y la videoconferencia continúe. Se adiciona el despliegue del video almacenado en el buffer por el usuario peer opuesto.
- El controlador se ubica en el servidor Web, está conformado por los patrones software de diseño *Proxy* y *Strategy*. El *Proxy* solicita a *Strategy* la acción a seleccionar en función del estado que le entrega el *Observer* del modelo y el *Observer* externo canal del subsistema externo.

En el subsistema externo tenemos un patrón de diseño *Observer* que permite validar el estado del canal de comunicación ante una alerta del *Observer* del almacenamiento temporal de la capa modelo del subsistema base.

El cliente Web realiza la solicitud de servicio de videoconferencia al *Proxy* del subsistema establecimiento, éste invoca al servidor quien solicita permiso al cliente Web para acceder al recurso de la cámara y micrófono del dispositivo móvil y la creación de un usuario. *Proxy* al recibir el usuario creado valida la información, para verificar si es nuevo o existe en la base de datos de usuario. Si es nuevo usuario lo crea y registra en

la base de datos. Si existe usuario, solicita a la controladora de base de datos la información del usuario para entregarle al servidor. Creados los clientes Web, *Proxy* espera que uno de ellos solicite el inicio de videoconferencia para pedir a servidor inicie establecimiento de sesión.

Una vez establecida la sesión, *Proxy* del subsistema sesión activa el *Observer* de modelo para que informe si existe algún cambio en el umbral del almacenamiento temporal. Dada la alerta por *Observer*, *Proxy* solicita confirmación mediante el monitoreo del *Observer* externo canal del subsistema externo sobre el estado del canal de comunicación.

Con un estado de interrupción, *Proxy* solicita a estrategia implantar el algoritmo interrupción que ejecuta dos actividades: solicitar el almacenamiento temporal de los flujos de audio y video emitidos por los clientes Web en los dispositivos inalámbricos y comunicar a sus vistas de esta situación mediante un mensaje: *Reconectando*.

Si el estado del canal indica que está conectado, *Proxy* pide a *Strategy* determinar acción y ejecuta el algoritmo reconexión para solicitar el restablecimiento de la sesión de videoconferencia, por lo que es llamado nuevamente el subsistema establecimiento. Una vez establecida la sesión, *Proxy* solicita al almacenamiento temporal de los clientes enviar flujos almacenados al almacenamiento temporal del servidor Web. El patrón *Adapter* convierte el flujo de audio y video y envía a los clientes Web opuestos para despliegue en vista de cada dispositivo móvil en modo bajo demanda.

5.1.2. Diseño arquitectónico y de despliegue

Se define la arquitectura propuesta que soporta los requisitos e implementa el modelo para video en tiempo real con control de interrupciones basado en patrones software. La Figura 5.3 presenta el diagrama de clases de este modelo.

La relación entre el modelo basado en patrones software y las clases que sido definidas en el diagrama de clases se describe a continuación:

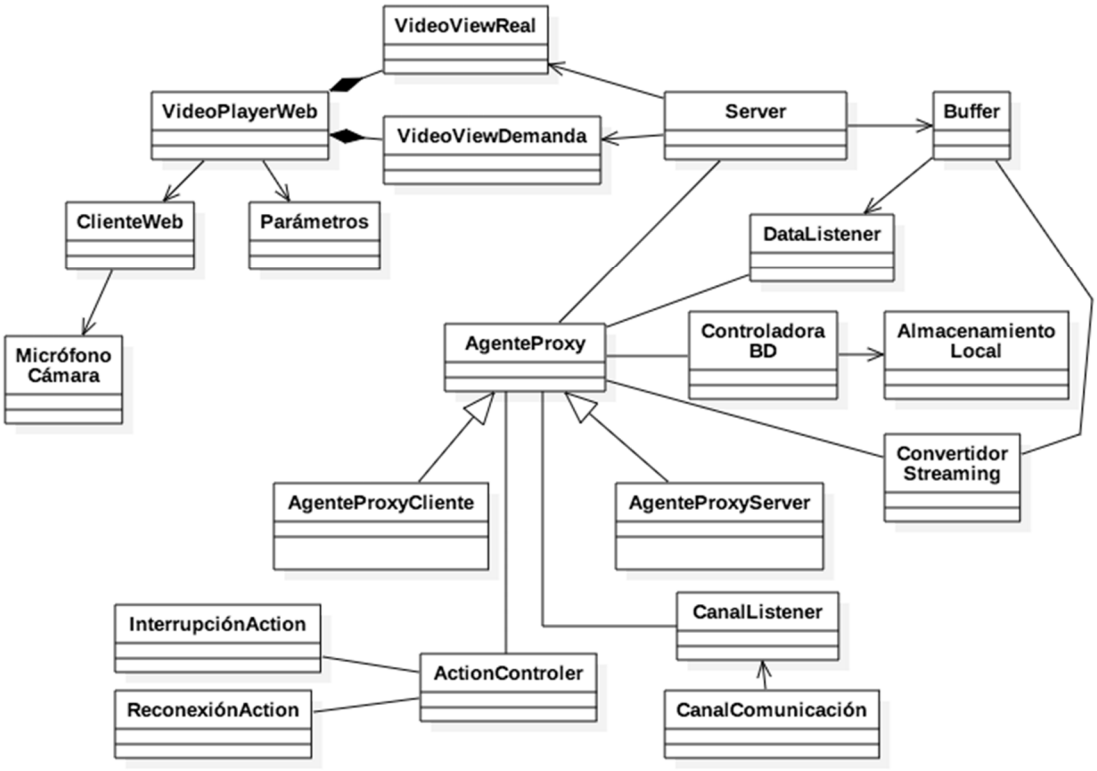


Figura 5.3: Diagrama de clases para modelo en tiempo real

En el subsistema establecimiento:

- *Modelo y patrón Adapter*: las clases que permiten el almacenamiento temporal y local y el convertidor video streaming que transcodifica la información son: *Server*, *Buffer*, *DataListener*, *ControladoraBD*, *AlmacenamientoLocal* y *ConvertidorStreaming*.
- *Vista y patrón Composite*: las clases que se ejecutan para lograr el establecimiento de la videoconferencia son: *ClienteWeb*, *VideoPlayerWeb*, *MicrófonoCámara* y *Parámetros*.
- *Controlador y patrón Proxy:* en este de define a la clase *AgenteProxy*, encargada de coordinar el establecimiento de la sesión de videoconferencia.

En el subsistema sesión:

- *Modelo y patrón Observer:* la clase que permiten el almacenamiento temporal y el observador que analiza constantemente el estado del buffer es *DataListener*.
- *Vista y patrón Composite:* las clases ejecutadas para lograr el despliegue de la videoconferencia en el dispositivo móvil son: *VideoPlayerWeb*, *VideoViewReal*, y *VideoViewDemanda*.
- *Controlador y patrón Proxy y Strategy:* el patrón *Proxy* el cual está representado por las clases *Proxy*, *AgenteProxyCliente* y *AgenteProxyServidor*. El patrón *Strategy* fortalece la capa de control implementado, mediante las clases: *InterrupciónAction* y *ReconexiónAction*.

En el subsistema externo tenemos el *Observer* externo que evalúa el estado del canal de comunicación mediante la clase *CanalListener*.

En la Figura 5.4 muestra el diagrama de secuencia, que me indica la cadena a seguir del modelo. Inicia con el cliente Web que define parámetros para pedir servicio de videoconferencia, aceptando acceso a recursos de micrófono y cámara. Se valida al usuario y se establece la sesión de videoconferencia. En este momento *Proxy* solicita estado de umbral del buffer; si existe alerta solicita al observador externo verificar el estado del canal de comunicación para confirmar si es un quiebre del canal de comunicación. *Proxy* solicita que se guarden los flujos capturados por la cámara y micrófono en un almacenamiento temporal. Una vez que se indique que el canal se ha restablecido, otras acciones son ejecutadas para restablecer la sesión y continuar con la videoconferencia. Finalmente son enviados al servidor los flujos almacenados por los clientes para que se transcodifique y sean entregados a sus pares correspondientes.

La Figura 5.5 presenta el diagrama de despliegue del modelo para video en tiempo real con control de interrupciones. Se compone de tres actores: dispositivo móvil, servidor de aplicaciones y servidor Web. El dispositivo móvil aloja al *ClienteWeb*, *VideoPlayerWeb*, *Parámetros*, *VideoView*, *Buffer*, *DataListener* y *AgenteProxyCliente*. Cada una de ellas permite interactuar con el despliegue de la videoconferencia y bajo demanda, definir parámetros, solicitar establecimiento de sesión y controlar el estado del umbral de almacenamiento

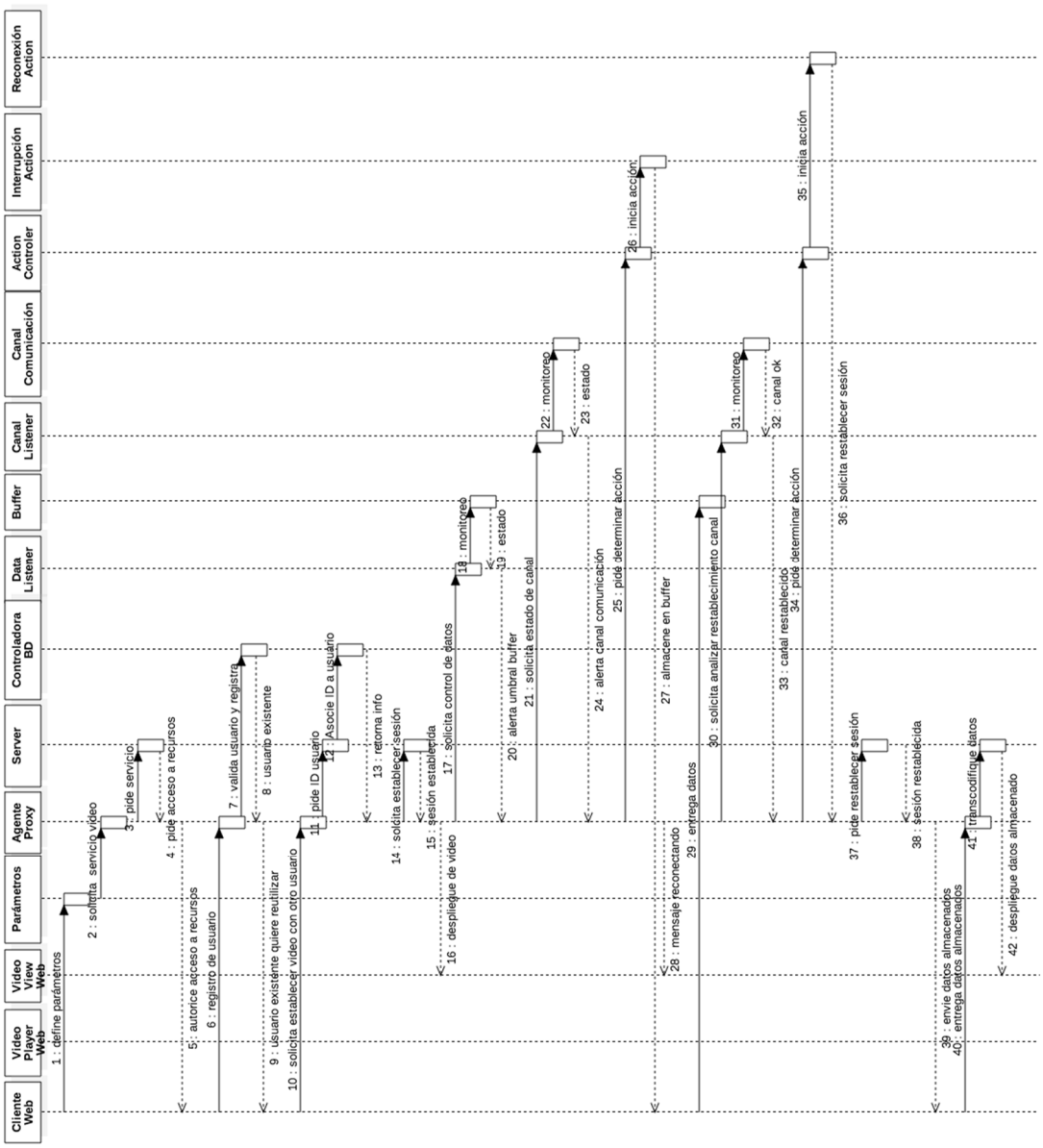


Figura 5.4: Diagrama de secuencia modelo para video en tiempo real

El servidor Web contiene a *Server*, *Buffer*, *AgenteProxyServer*, *Adaptador*, *AlmacenamientoLocal*, *ActionControler*, *InterrupciónAction* y *ReconexiónAction*. Las funciones que implementan son la administración de usuarios, establecimiento de la sesión, control de estado de umbral de buffer y canal de comunicación, transcodificación de flujos de datos, control de interrupción mediante acción de eventos y restablecimiento de la sesión de videoconferencia.

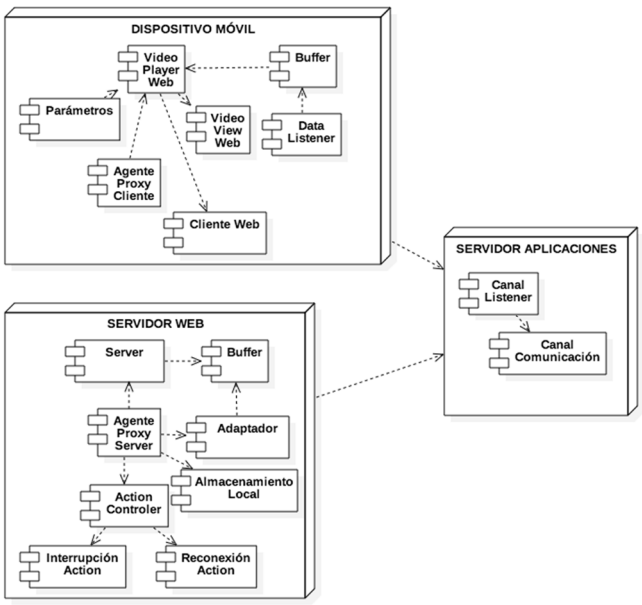


Figura 5.5: Diagrama de despliegue modelo para video en tiempo real

El servidor de aplicaciones contiene a *CanalListener* y *CanalComunicación*, que se encarga de observar al canal de comunicación y enviar el estado del canal al *AgenteProxy*.

5.2. Modelo matemático de rendimiento

El modelo para video en tiempo real con control de interrupciones mantiene el modelo base. Esto quiere decir que el mecanismo base basada en arquitectura proxy se implementa en este modelo.

Este modelo presenta incrementales de funcionalidad al implantar tres subsistemas. Uno de ellos permite establecer la sesión para video en tiempo real, este es uno de los nuevos actores que debemos contemplar en el modelo.

Partiendo del modelo matemático del capítulo 3 tenemos que dada su ecuación, ahora debemos considerar el tiempo de establecimiento de la sesión. El modelo considera un tiempo de inicio de la sesión de video streaming, entonces ahora nosotros lo que tenemos es que el tiempo de inicio es igual al tiempo de establecimiento que plantea el modelo para video en tiempo real.

Entonces tenemos:

$$T_{INICIO} = T_{ESTABLECMIENTO}$$

La ecuación del modelo del capítulo 3 es:

$$T_{EJECUCION} = T_{INICIO_{FLUJO}} + \sum_{i=1}^{n} T_{DATO_i} + \sum_{i=1}^{n} T_{INTERRUPCION_i} + T_{DATOn}$$

Donde:

T_{INICIO} es el tiempo de inicio de solicitud del servicio de video streaming.

$\sum_{i=1}^{n}(T_{DATO_i})$ es el tiempo total de T_{DATO} transmitidos una sola vez.

$\sum_{i=1}^{n} T_{INTERRUPCION_i}$ es el tiempo total de interrupciones.

T_{DATO_n} es tiempo del último T_{DATO} transmitido.

Reemplazado la igualdad planteada tenemos:

$$T_{EJECUCION} = T_{ESTABLECIMIENTO} + \sum_{i=1}^{n} T_{DATO_i} + \sum_{i=1}^{n} T_{INTERRUPCION_i} + T_{DATOn}$$

Como la transmisión es en tiempo real, T_{DATO} es cero, tenemos:

$$T_{EJECUCION} = T_{ESTABLECIMIENTO} + \sum_{i=1}^{n} T_{DATO_i} + \sum_{i=1}^{n} T_{INTERRUPCION_i}$$

Ahora si tenemos una interrupción, está genera que se almacene la información de video en tiempo real en un buffer. Quiere decir que tenemos un flujo que devolverá el mecanismo una vez que cancela la interrupción. Por lo tanto bajo esta consideración tenemos que incrementar un tiempo de visualización del video bajo demanda.

Con esta consideración tenemos la siguiente ecuación:

$$T_{EJECUCION} = T_{ESTABLECIMIENTO} + \sum_{i=1}^{n} T_{DATO_i} + \sum_{i=1}^{n} T_{INTERRUPCION_i} + T_{VIDEOBAJODEMANDA}$$

Donde:

$T_{ESTABLECIMIENTO}$ es el tiempo de establecimiento de la sesión.

$\sum_{i=1}^{n}(T_{DATO_i})$ es el tiempo total de T_{DATO} transmitidos.

$\sum_{i=1}^{n} T_{INTERRUPCION_i}$ es el tiempo total de interrupciones.

$T_{VIDEOBAJODEMANDA}$ es el tiempo total de flujo almacenado.

Se realiza un escenario de pruebas para este modelo con las siguientes consideraciones:

- El tiempo video bajo demanda tendrá un máximo de almacenamiento para 15 minutos.
- Como es video en tiempo real se analiza dos escenarios, uno con interrupciones muy pequeñas y otro con interrupciones muy grandes.

Los datos experimentales para el mejor escenario se describen en la tabla 5.7. Los datos experimentales para el peor escenario se describen en la tabla 5.8.

Tabla 5.7: Datos experimentales mejor escenario

MEJOR ESCENARIO										
PRUEBA	Testablecim	Tflujo1	Tflujo2	Tflujo3	Tflujo4	Tint1	Tint2	Tint3	Tvideobajod	Tejecucion
1	2,08	141,43	125,3	138,77	494,5	12,28	13,16	10,9	900	938,42
2	2,11	118,8	181,13	110,04	490,03	11,86	11,55	14,83	900	940,35
3	2,21	171,96	113,96	129,26	484,82	11,83	13,41	9,06	900	936,51
4	2,43	111,8	122,18	154,43	511,59	12,21	10,9	16,06	900	941,6
5	2,8	161,32	150,2	134,35	454,13	11,11	9,54	8,98	900	932,43
6	2,76	130,75	128,65	132,79	507,81	11,05	12,23	13,25	900	939,29

Tabla 5.8: Datos experimentales peor escenario

PEOR ESCENARIO										
PRUEBA	Testablecim	Tflujo1	Tflujo2	Tflujo3	Tflujo4	Tint1	Tint2	Tint3	Tvideobajod	Tejecucion
1	2,23	123,34	134,21	225,41	417,04	73,43	66,34	70,89	900	1112,89
2	2,18	345,34	115,08	51,98	387,6	85,23	72,56	69,44	900	1129,41
3	2,7	76,34	152,16	234,81	436,69	76,78	66,28	70,38	900	1116,14
4	2,43	84,34	197,79	223,81	394,06	79,23	89,46	72,84	900	1143,96
5	2,28	144,01	147,45	217,51	391,03	81,12	81,29	71,78	900	1136,47
6	2,1	88,74	130,88	83,13	597,25	76,78	83,76	80,54	900	1143,18

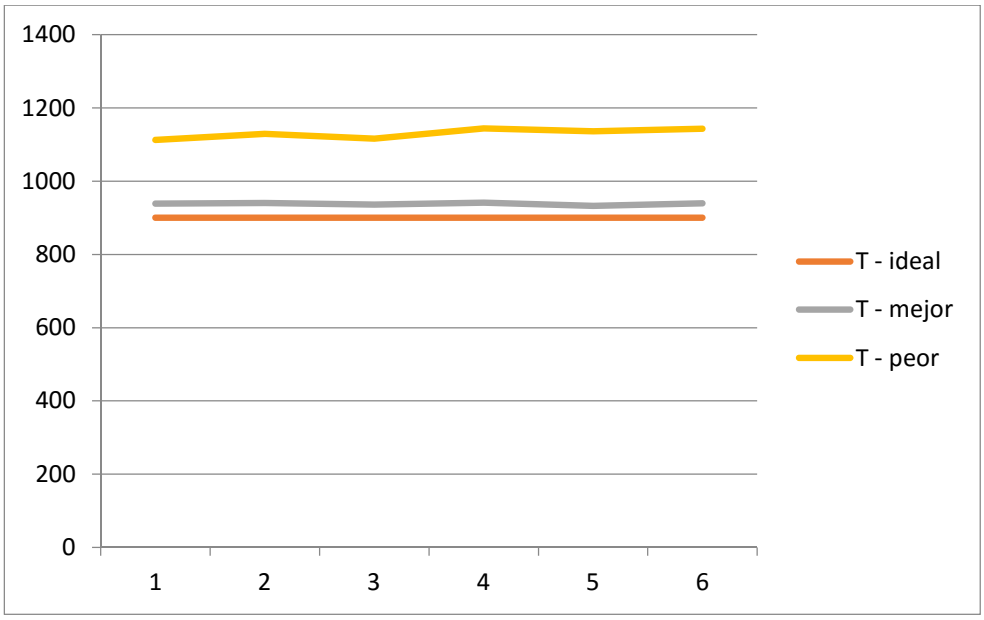


Figura 5.6: Tiempos de ejecución en escenarios

Se realizó la gráfica de los escenarios (Figura 5.6) y tenemos las siguientes conclusiones:

- Mejora notablemente el tiempo de ejecución debido a que no tenemos tiempos acumulados por retransmisiones de flujos.
- En este modelo no influye si las interrupciones se dan al inicio o al final del video.
- Es importante considerar que el modelo se sustenta en utilizar un almacenamiento temporal en el servidor de aplicaciones, para almacenar los flujos que no se pueden enviar al cliente por interrupción del canal de comunicación. Entonces ahora es necesario analizar el coste que me implicaría y si impacta en el modelo.

La cantidad de memoria requerida en este escenario lo determina el tiempo video bajo demanda de 900 segundos, y con una capacidad del canal ADSL de 10Mbps es:

$$Capacidad\ memoria = Capacidad\ canal \times Tiempor\ interrupción$$

$$Capacidad\ memoria = 10 \times 900$$

$$Capacidad\ memoria = 9000\ Mbits$$

$$Capacidad\ memoria = \frac{9000\ Mbits}{8} = 1125\ MB$$

Entonces se concluye que la capacidad necesitada no afecta en coste al modelo. Además tenemos alternativas hoy en día como almacenamiento locales muy pero muy grandes, orden de los TB, y almacenamientos en la nube sin coste por el orden de los GB. Es importante exponer que lo que se busca es mantener una buena percepción del usuario, por lo tanto el importe que tenemos que pagar por el coste de almacenamiento es relativamente mínimo.

5.3. Análisis de la implantación experimental

La arquitectura mostrada en la Figura 5.7 propone una solución de control de interrupciones, basada en la implementación de proxies con agentes de software, para la comunicación en tiempo real Web peer-to-peer dada una interrupción en el canal de comunicación inalámbrica.

Entre (1) y (8) se establece el proceso de comunicación y entre (9) y (10) se realiza el proceso de petición de estadísticas de comunicación y eventos que permiten definir si es interrupción normal o interrupción por alguna interrupción en el canal inalámbrico de comunicación entre pares.

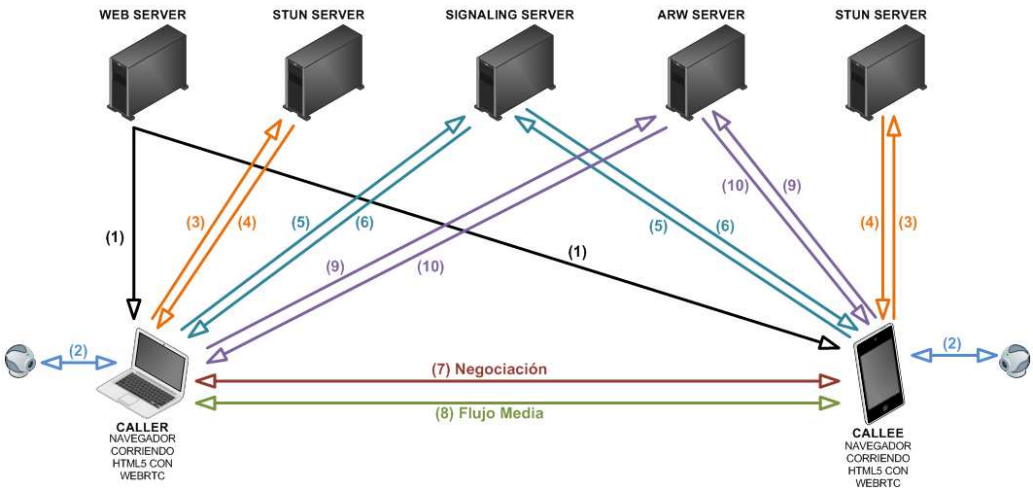


Figura 5.7: Visión general de la arquitectura ARW

5.3.1. Proyección de patrones software en los diagramas de diseño

El diseño se realiza en base a los requisitos planteados, obteniéndose el diagrama de clases, de secuencia y despliegue, como se puede observar en las Figuras 5.8 – 5.10.

La relación entre el modelo basado en patrones software y las clases que sido definidas en el diagrama de clases se describe a continuación:

En el subsistema establecimiento:

- *Modelo y patrón Adapter*: las clases que permiten el almacenamiento temporal y local y el convertidor video streaming que transcodifica la información son: *Server*, *PeerServer*, *PeerJs* y *usuario*.
- *Vista y patrón Composite*: las clases que se ejecutan para lograr el establecimiento de la videoconferencia son: *Usuario*, *Video*, *InputText*, y *Button*.
- *Controlador y patrón Proxy:* en este de define a la clase *ConexionBd*, *SocketServer* y *SocketIO* encargada de coordinar el establecimiento de la sesión de videoconferencia.

En el subsistema sesión:

- *Modelo y patrón Observer:* la clase que permiten el almacenamiento temporal y el observador que analiza constantemente el estado del buffer es *Comunicacion*.
- *Vista y patrón Composite:* las clases ejecutadas para lograr el despliegue de la videoconferencia en el dispositivo móvil son: *Usuario*, *Video*, *TiempoComunicacion* y *Button*.
- *Controlador y patrón Proxy y Strategy:* el patrón *Proxy* el cual está representado por las clases *SocketServer*, *SocketCliente* y *ConexionBd*. El patrón *Strategy* fortalece la capa de control implementado, mediante la clase: *SocketIO*.

En el subsistema externo tenemos el observador externo que evalúa el estado del canal de comunicación mediante la clase *Comunicacion*.

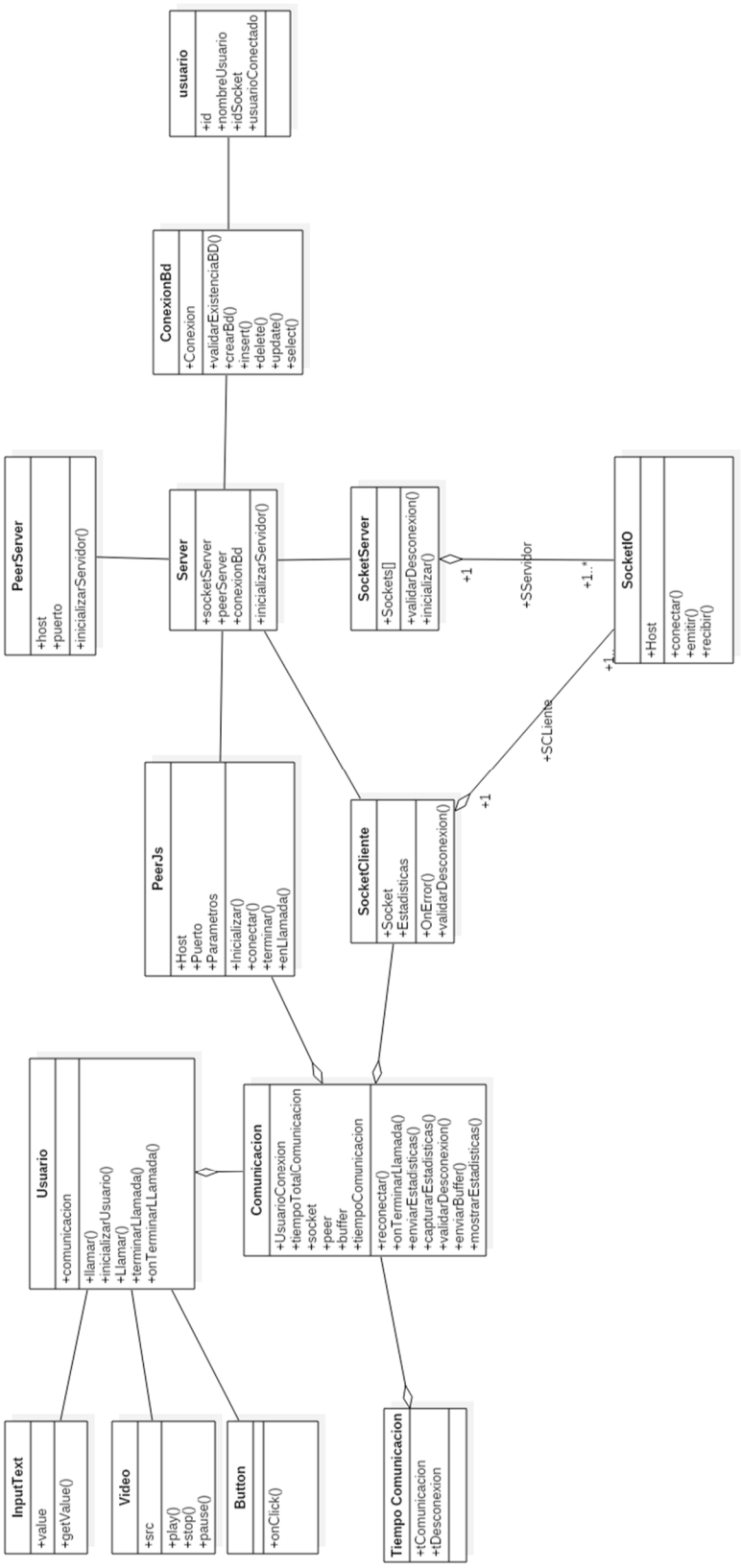


Figura 5.8: Diagrama clase solución video en tiempo real

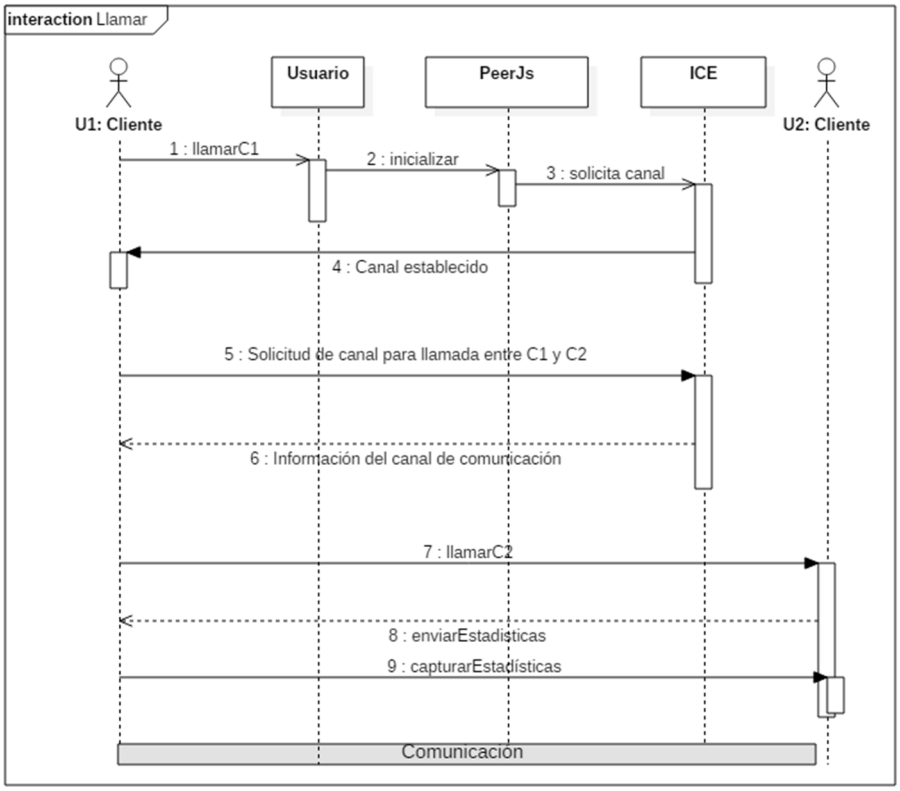


Figura 5.9: Diagrama de secuencia establecimiento

En la Figura 5.9 y 5.10 se muestran los diagramas de secuencia de establecimiento y reconexión, que me indica la cadena a seguir del modelo. Inicia con el cliente Web que define parámetros para pedir servicio de videoconferencia, aceptando acceso a recursos de micrófono y cámara que solicita *PeerJs*.

Se valida al usuario y se establece la sesión de videoconferencia mediante *Ice*. En este momento *Proxy* solicita estado de umbral del buffer; si existe alerta solicita al *Observer* externo *Comunicación* verificar el estado del canal de comunicación para confirmar si es un quiebre del canal de comunicación. *Proxy* solicita que se guarden los flujos capturados por la cámara y micrófono en un almacenamiento temporal. Una vez que se indique que el canal se ha restablecido, otras acciones son ejecutadas para restablecer la sesión y continuar con la videoconferencia.

Finalmente son enviados al servidor los flujos almacenados por los clientes para que se transcodifique y sean entregados a sus pares correspondientes.

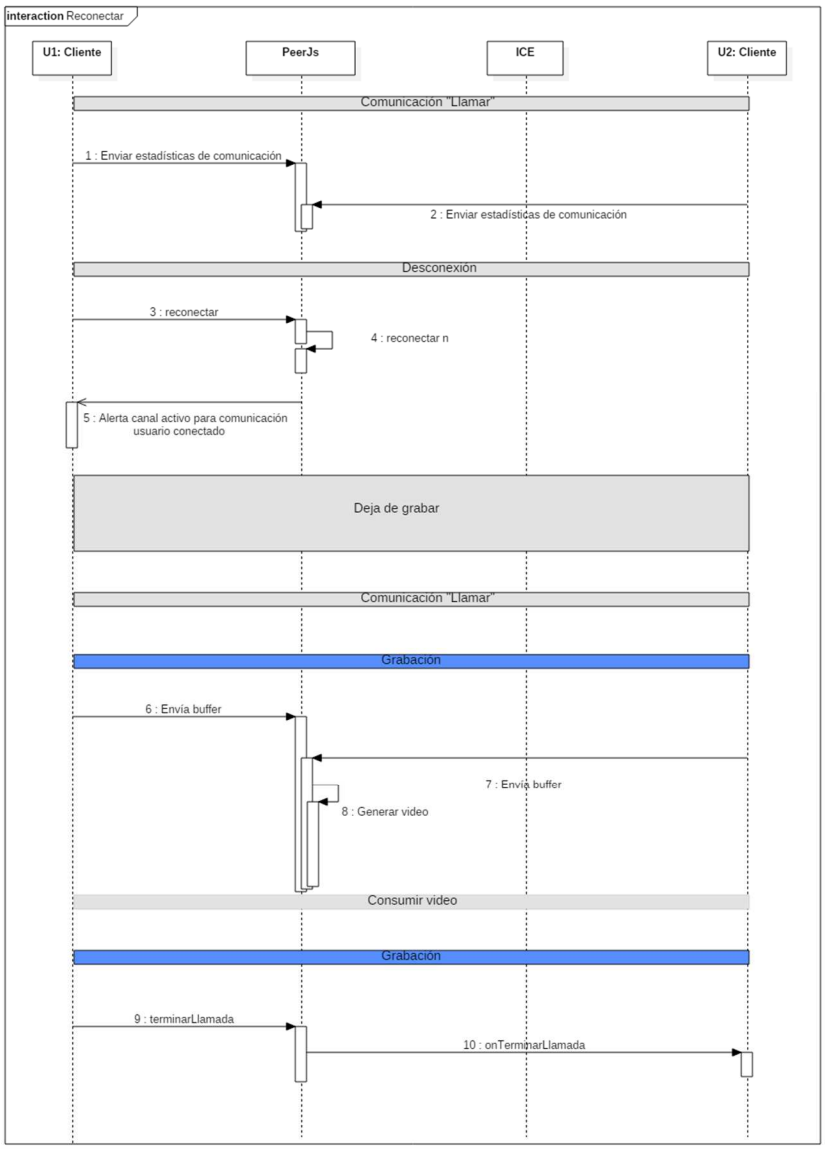


Figura 5.10: Diagrama de secuencia reconexión

La Figura 5.11 presenta el diagrama de despliegue del modelo para video en tiempo real con control de interrupciones. Se compone de cuatro actores: servidor, cliente, base de datos e IceServer.

El cliente aloja al WebRTCApp, Usuario, Comunicacion, HTML5, Chrome, Cliente PeerJs, Cliente Socket y Estadisticas.

Cada una de ellas permite interactuar con el despliegue de la videoconferencia y bajo demanda, definir parámetros, solicitar establecimiento de sesión y controlar el estado del umbral de almacenamiento

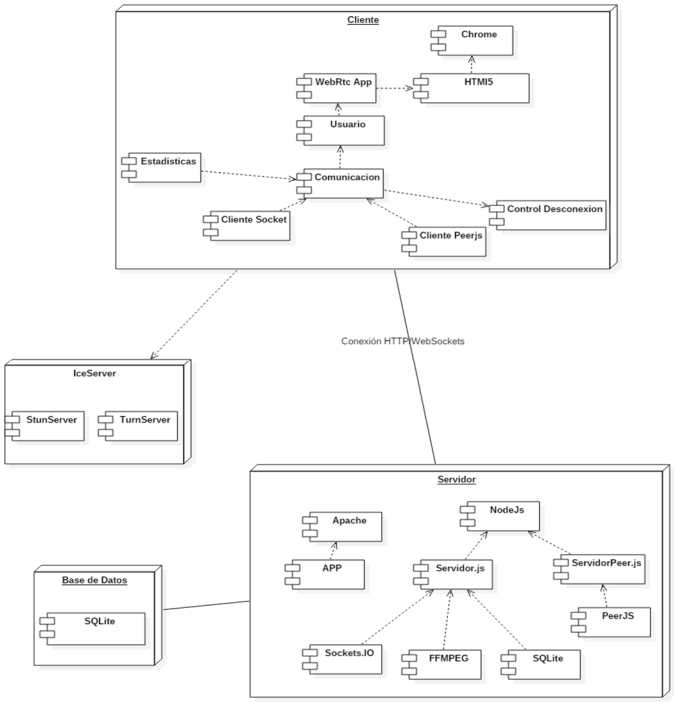


Figura 5.11: Diagrama de despliegue

El servidor contiene a Apache, NodeJs, Servidor, ServidorPeer, PeerJs, ffmpeg, SQlite y SocketIO. Las funciones que implementan son la administración de usuarios, establecimiento de la sesión, control de estado de umbral de buffer y canal de comunicación, transcodificación de flujos de datos, control de interrupción mediante acción de eventos y restablecimiento de la sesión de videoconferencia. Además se detalla al servidor Ice y la base de datos, que son parte del servidor.

5.3.2. Proyección del diagrama de secuencia y despliegue

La Figura 5.12 describe la arquitectura ARW (Automatic Reconnection WebRTC), precisa tres componentes, un servidor Web que procesa la aplicación realizando conexiones bidireccionales y síncronas o asíncronas con el usuario, un servidor WebRTC que permite establecer una comunicación peer-to-peer en una aplicación del navegador y un servidor ARW que controla la interrupción y reconexión automática de la comunicación peer-to-peer ante una interrupción. La arquitectura se encuentra desarrollada en JavaScript y utiliza como intérprete a Node.js.

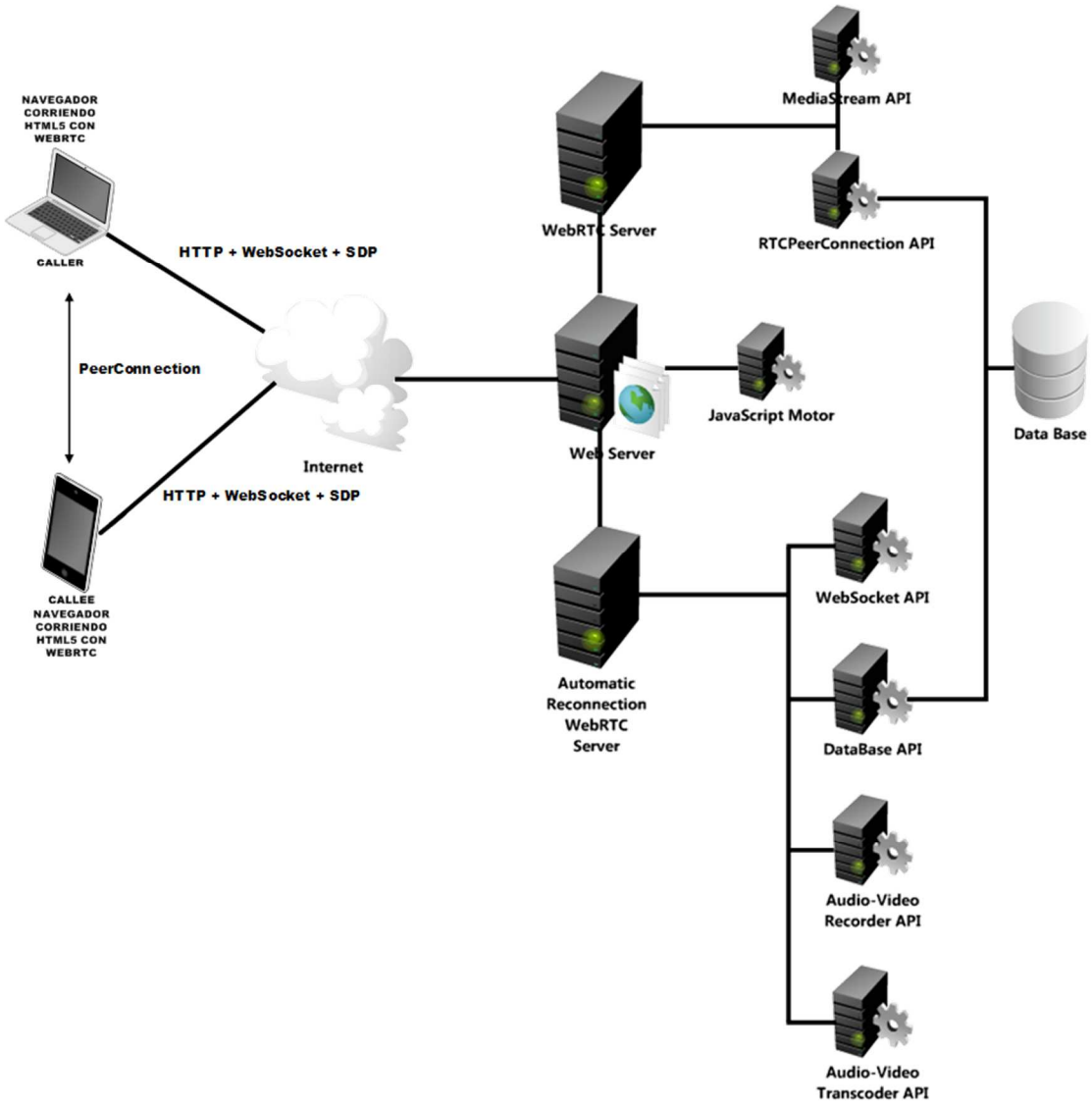


Figura 5.12: Arquitectura ARW

ARW implementa su mecanismo mediante tres APIs:

- WebSocket, provee una comunicación bidireccional basada en eventos en tiempo real de tipo cliente/servidor. Utiliza Socket.IO, librería de JavaScript para Node.js, para establecer los Websockets para proceso de petición de estadísticos de comunicación y eventos que permitan definir si es interrupción normal o interrupción por alguna interrupción en el canal inalámbrico de comunicación entre pares.

- DataBase, proporciona el servicio de almacenamiento de información del sistema de administración de usuarios y los datos media (capturados por la cámara y micrófono) relacional de la comunicación peer-to-peer. Utiliza SQLite, como motor de base de datos, para almacenar una base de datos completa en un único archivo de disco de plataforma cruzada y multiplataforma.
- Audio-Video Record, permite la grabación de media en navegadores Web inmediatamente si el agente software del par valida una interrupción por interrupción en su canal inalámbrico. Utiliza RecordRTC, librería de grabación de media basada en JavaScript para navegadores Web con soporte WebRTC.
- Audio-Video Transcoder, realiza la transcodificación del audio y video en tiempo real una vez que la comunicación del canal inalámbrico se recuperó y envía a almacenar en la base de datos relacional

La plataforma inicia estableciendo una comunicación entre el usuario y el servidor mediante el API de Web sockets por el puerto 9001, a través del cual realizan un intercambio de información, además de solicitar al usuario los permisos necesarios para acceder hacia los recursos de la cámara y micrófono del dispositivo, también configura los parámetros para el enlace con el servidor de WebRTC.

Posteriormente se solicita el registro de usuarios en el cual se realiza el siguiente proceso: se valida si el usuario había hecho antes uso de la aplicación, con lo cual se le brinda la opción de registrar un nuevo usuario o seguir utilizando el que ya tenía disponible.

Mediante este proceso se enlaza el nombre de usuario con el ID único *Universally Unique Identifier* (*UUID*) que es un HASH único que le provee el servidor de WebRTC, y con el cual se establece la comunicación entre los usuarios, adicionalmente relaciona el socket establecido inicialmente con el nuevo usuario, la plataforma valida que no exista más de un usuario con el mismo ID ya que esto ocasionaría problemas al momento de la comunicación. Se debe indicar que socket en el momento que se conecta asocia el ID y nombre de usuario al ID único que posee el socket.

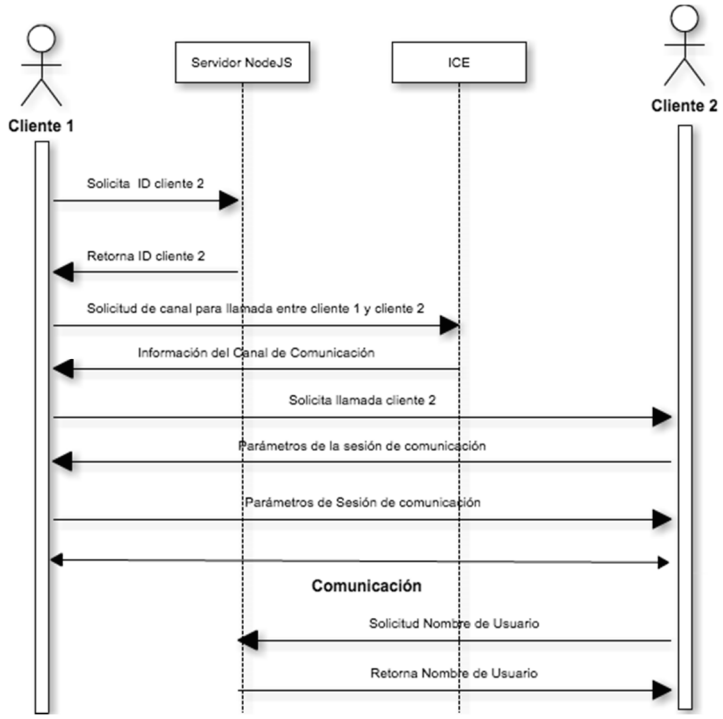


Figura 5.13: Establecimiento de la comunicación

Pasado el proceso de registro se presenta una pantalla en la cual se solicita ingresar un nombre de usuario con el cual se desee establecer una comunicación. Al ingresar un nombre de usuario lo primero que se realiza es una solicitud hacia el servidor para que responda el id correspondiente al usuario para establecer la comunicación a través de WebRTC. Con el ID inicia el proceso de comunicación, el cual solicita al servidor ICE la ruta por la cual se va a establecer la comunicación entre los dos usuarios, utilizando los servidores STUN y TURN. El servidor por defecto que se utiliza IceServer es STUN: stun.l.google.com: 19302 (Figura 5.13).

Una vez iniciada la comunicación entre dos usuarios la plataforma captura las estadísticas de la comunicación proporcionadas por WebRTC, las mismas que son enviadas hacia el servidor con lo cual se valida que la comunicación este activa y no se han dado errores durante la misma, este proceso es realizado cada 3 segundos a través de los Web sockets. El objeto de WebRTC en este caso Peerjs posee una propiedad que va almacenando las estadísticas de la comunicación capturadas por la API stats. El envío se lo realiza a través de la comunicación de Web sockets. WebRTC-internals es el medio para analizarlas y mediante un método en Web sockets se dispara cuando se pierde la

comunicación. Además el objeto Peerjs tiene un evento que detecta la finalización de una llamada. El servidor valida si la interrupción fue realizada por el usuario, con la comparación de la información de los métodos indicados anteriormente.

Cuando se produce un error de comunicación con el servidor y el usuario se encuentra en una comunicación, la plataforma inicia un proceso de validación de interrupción. En el cual ambos usuarios intentan comunicarse con el servidor para validar si la interrupción se dio por uno de los usuarios (se valida que la comunicación WebRTC presenta un evento de cierre pero que no fue solicitado por ninguno de los usuarios, solicitando una confirmación hacia el servidor del corte de la comunicación) o por errores durante la comunicación (se valida que la comunicación ha sido cerrada y además que se tenga comunicación con el servidor enviándole señales a través de Web sockets).

En caso de presentarse el segundo evento mencionado, el servidor notifica a los usuarios que deben iniciar con un proceso de reconexión de la comunicación y almacenamiento de un buffer de la información. Existe la posibilidad de no restablecimiento de la comunicación por los tiempos de respuesta del servidor Peerjs e Ice Server.

Si uno de los clientes perdió la comunicación tanto con el servidor (WebSockets) como con el otro usuario (Comunicación WebRTC), automáticamente inicia un proceso alterno de intento de reconexión con el servidor y almacenamiento del buffer, hasta cuando recibe una respuesta afirmativa del servidor con lo cual inicia un proceso de reconexión de comunicación, instanciando un nuevo objeto (PeerConnection por el puerto 9000) con lo cual reenvía sus nuevos datos hacia el servidor, el cual notifica al otro cliente que inicia un proceso de reconexión y solicita a ambos clientes el buffer almacenado durante la interrupción para poder generar un archivo que intercambia con el otro usuario (Figura 5.14).

Una vez que el usuario recibe la notificación del servidor empieza un nuevo proceso de establecimiento de comunicación debido a la instanciación del nuevo objeto, lo que asegura que se restablezca la comunicación por un nuevo canal, que no contenga errores.

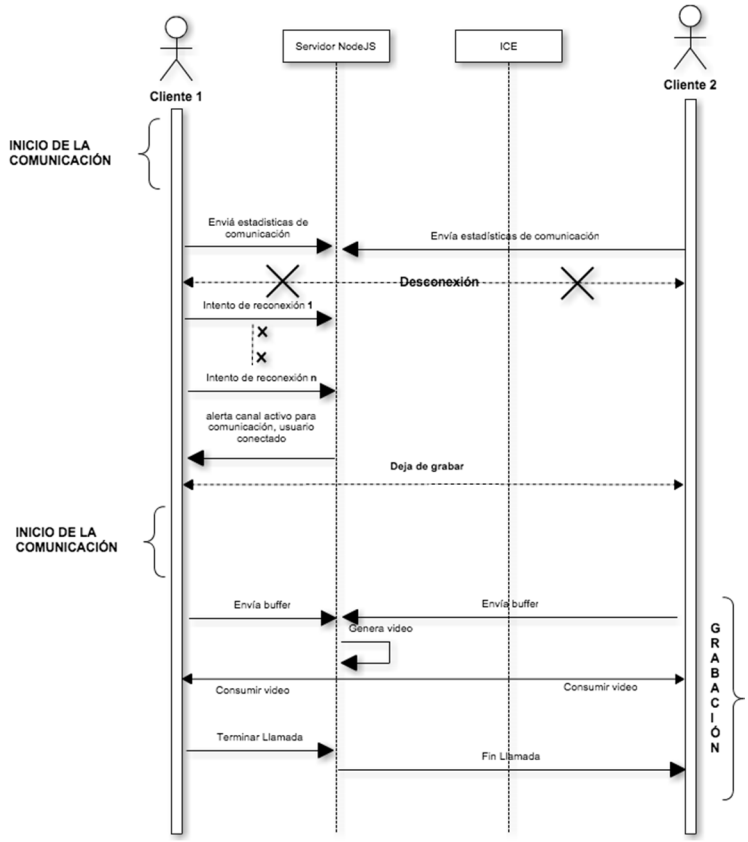


Figura 5.14: Comunicación y reconexión

En el caso de que únicamente se perdió la comunicación WebRTC entre los usuarios pero se mantuvo la comunicación con el servidor ambos usuarios inician con el proceso de reconexión y almacenamiento del buffer, instanciando nuevos objetos de comunicación para ambos clientes. Una vez que ambos clientes finalizaron este proceso envían sus nuevos datos hacia el servidor, indica a los clientes que inicia el proceso de reconexión solicitando el buffer almacenado y empieza un nuevo proceso de establecimiento de comunicación.

5.3.3. Resultados experimentales

La arquitectura desarrollada está implementada en una máquina virtual con 4 vCPU(s) de procesamiento en topología de 2 sockets con 2 cores por socket, memoria RAM de 8GB y sistema operativo Windows. Como clientes o pares, se utilizaron: una iMac con procesador Intel core i5 de 3.1GHz, memoria RAM de 4GB, sistema operativo OS X

Yosemite v10.10.2 y tarjeta inalámbrica Airport Extreme Atheros 9380 IEEE 802.11a/b/g/n; una MacBook Pro con procesador Intel core i7 de 2.0GHz, memoria RAM de 4GB, sistema operativo OS X Yosemite v10.10.2 y tarjeta de red inalámbrica Airport Extreme Broadcom BCM43xx IEEE 802.11a/b/g/n; un dispositivo móvil Samsung Galaxy S4 modelo GT-i9500 con procesador Quad-Core 1,6GHz ARM Cortex-A15, memoria RAM de 2GB, sistema operativo Android 5.0.1 Lollipop y IEEE 802.11a/b/g/n; un dispositivo móvil Samsung Galaxy modelo GT-i9192 con procesador Dual-Core 1,7GHz Qualcomm Snapdragon Cortex-A15, memoria RAM de 1,5GB, sistema operativo Android 4.4.2 Kitkat y IEEE 802.11a/b/g/n. Además, se utilizó una red inalámbrica IEEE 802.11a/b/g/n con velocidad de enlace promedio de 150 Mbps, una latencia promedio de 22ms y un poder de emisión de radio promedio de 32,5dBm.

Las pruebas se realizaron en dos escenarios:

- Computador – Computador
- Dispositivo móvil – Dispositivo móvil

Se establecieron los siguientes parámetros para los escenarios de pruebas:

- Tiempo de duración de la comunicación de 2 a 3 minutos
- Tiempo de interrupción de 10 a 15 segundos, número de interrupciones 3.

Comunicación entre dos usuarios (Computador – Computador)

La Figura 5.15 muestra la comunicación entre dos usuarios, ambos permitieron el acceso a la cámara y micrófono, uno de ellos registra hacer llamada y se establece la comunicación entre los dos usuarios con una calidad alta en la comunicación. Se visualiza dos elementos de video uno del usuario remoto y otro del usuario local.

Comunicación entre dos usuarios (Dispositivo móvil – Dispositivo móvil)

La Figura 5.16 muestra la comunicación entre dos usuarios, ambos permitieron el acceso a la cámara y micrófono, uno de ellos registra hacer llamada y se establece la comunicación entre los dos usuarios con una calidad media en la comunicación. Se visualiza dos elementos de video uno del usuario remoto y otro del usuario local.

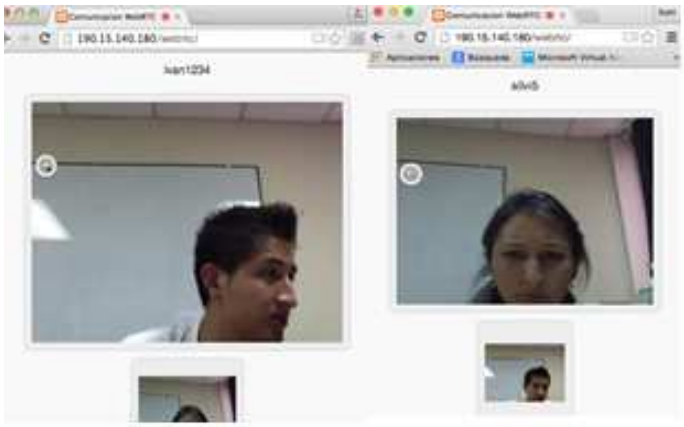


Figura 5.15: Comunicación entre Computador-Computador

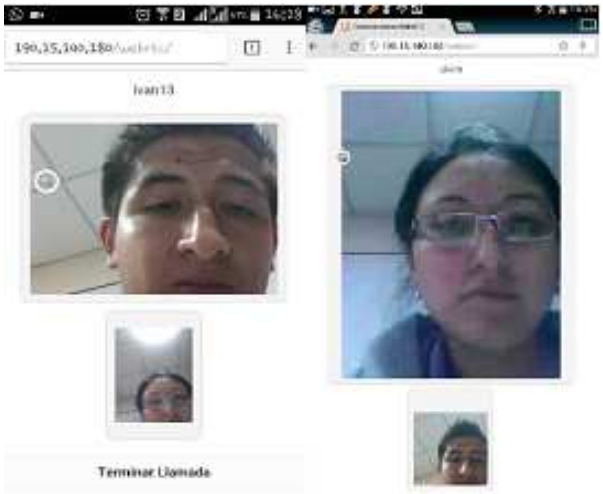


Figura 5.16: Comunicación entre Dispositivo móvil-Dispositivo móvil

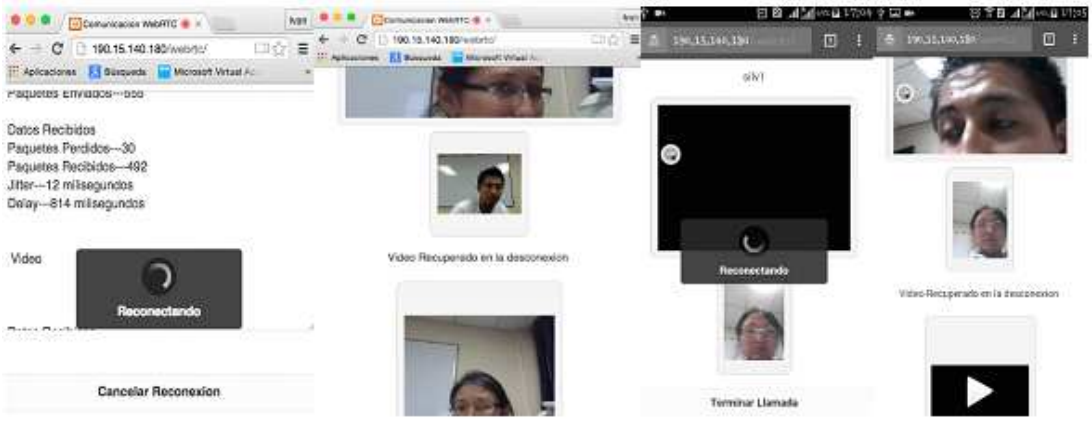


Figura 5.17: Interrupción y reconexión entre dos usuarios

Reconexión automática y recuperación de video

La Figura 5.17 muestra la comunicación entre dos usuarios, que presenta una interrupción y posterior reconexión automática de la comunicación y recuperación del video capturado durante la interrupción. Se restablece la comunicación entre los usuarios en todos los escenarios y se visualiza el video recuperado respectivo. Se realizó la experimentación y toma de la muestra para los dos escenarios que se anexa, para valorar los parámetros de jitter, retraso y pérdida de paquetes.

Escenario Computador – Computador

En la Tabla 5.9 y la Figura 5.18-5.23 se muestran los resultados para una comunicación inicial, una interrupción y una reconexión. El análisis se realizó al canal de audio como al de video. De los resultados y aplicando la recomendación de la ITU-T G.114 [ITU, 2003] se concluye que el mecanismo de reconexión automática de la arquitectura ARW no afecta a la comunicación debido a que en ambos casos no superan los valores indicados en G.114 en jitter y Retraso.

La Figura 5.18 indica el retraso y jitter obtenidos en la comunicación inicial de la sesión para el canal de video. La Figura 5.19 indica el retraso y jitter obtenidos en la comunicación inicial de la sesión para el canal de audio. La Figura 5.20 indica el retraso y jitter obtenidos dada la interrupción de la sesión para el canal de video. La Figura 5.21 indica el retraso y jitter obtenidos dada la interrupción de la sesión para el canal de audio. La Figura 5.22 indica el retraso y jitter obtenidos dada la reconexión de la sesión para el canal de video. La Figura 5.23 indica el retraso y jitter obtenidos dada la reconexión de la sesión para el canal de audio.

Tabla 5.9: Resultados escenario Computador-Computador

ESCENARIO COMPUTADOR – COMPUTADOR		
	Computador 1	Computador 2
COMUNICACIÓN INICIAL		
Canal de Video		
Jitter	25ms - 5ms	15ms - 25ms - 15ms

Retraso	50ms – 30ms	100ms – 50ms
Canal de Audio		
Jitter	5ms – 0ms	5ms – 0ms
Retraso	150ms – 60ms	150ms – 55ms
INTERRUPCIÓN		
Canal de Video		
Jitter	25ms - 12ms	25ms - 8ms
Retraso	60ms – 45ms	65ms – 65ms
Canal de Audio		
Jitter	12ms – 12ms	13ms – 13ms
Retraso	130ms – 130ms	105ms – 105ms
RECONEXION		
Canal de Video		
Jitter	25ms - 10ms	25ms – 11ms
Retraso	100ms – 65ms	100ms – 60ms
Canal de Audio		
Jitter	10ms – 0ms	12ms – 0ms
Retraso	200ms – 55ms	200ms – 5ms

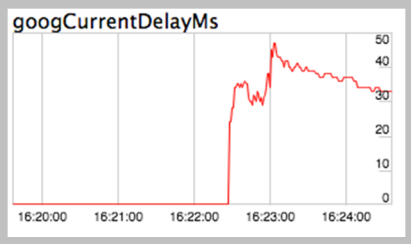
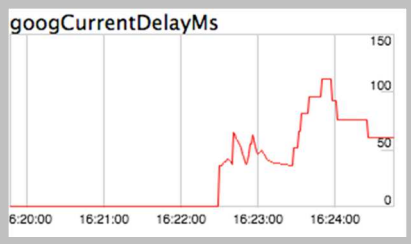
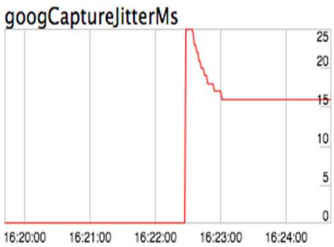
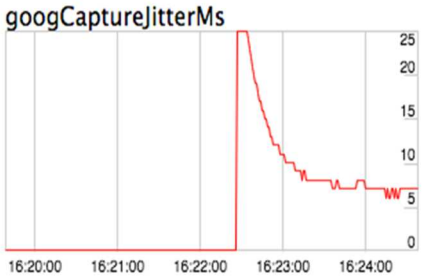
	Computador 1	Computador 2
Retraso		
Jitter		

Figura 5.18: Comunicación inicial – Canal de Video

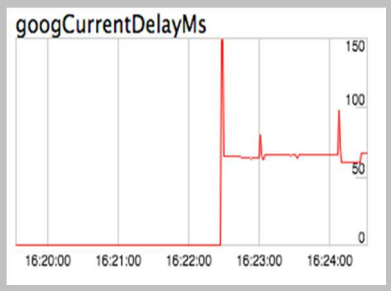
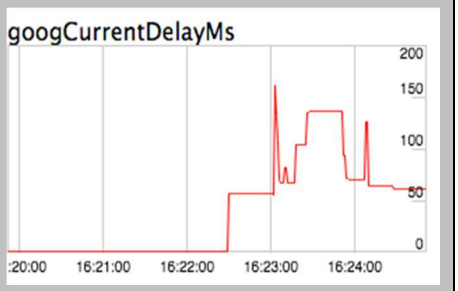
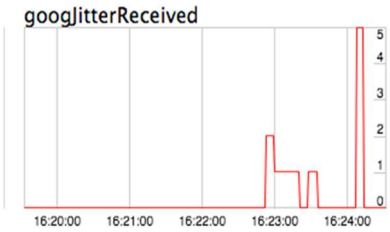
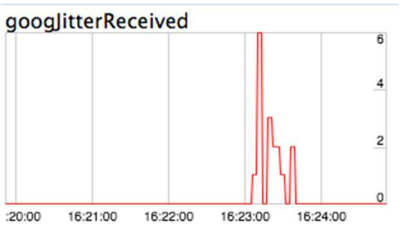
Audio	Computador 1	Computador 2
Retraso		
Jitter		

Figura 5.19: Comunicación inicial – Canal de Audio

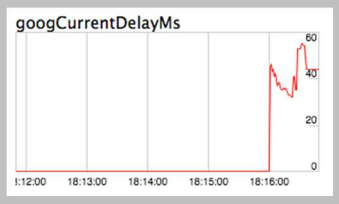
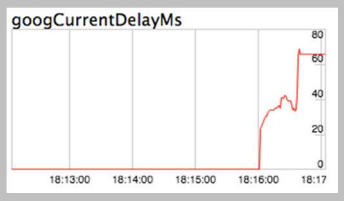
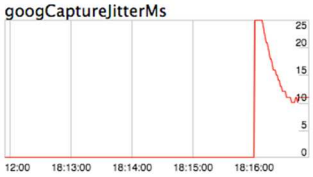
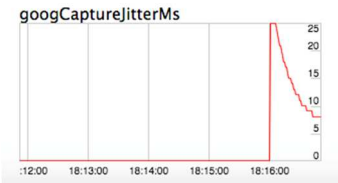
	Computador 1	Computador 2
Retraso		
Jitter		

Figura 5.20: Interrupción – Canal de Video

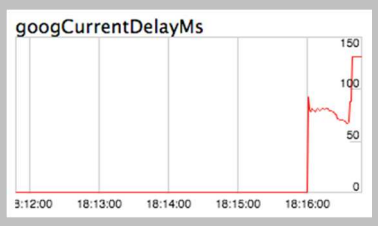
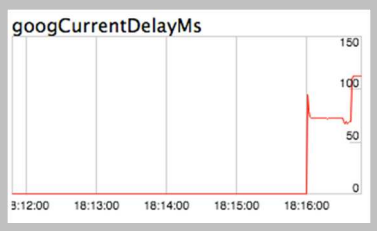
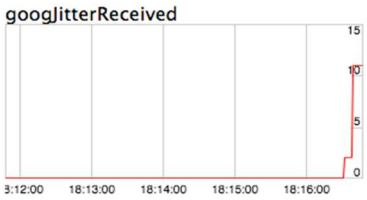
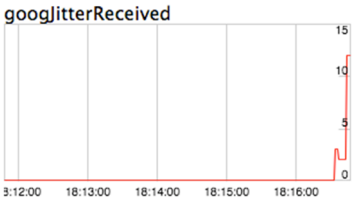
	Computador 1	Computador 2
Retraso		
Jitter		

Figura 5.21: Interrupción – Canal de Audio

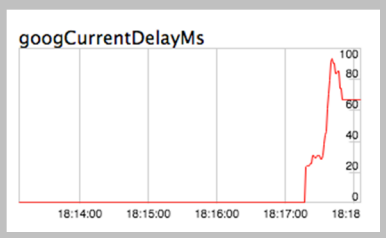
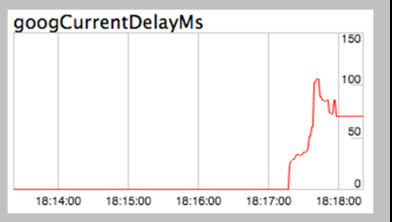
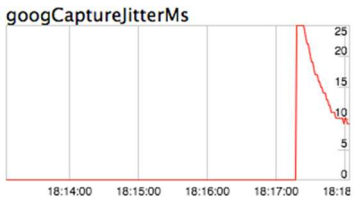
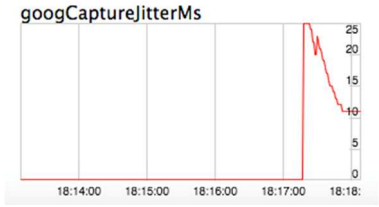
	Computador 1	Computador 2
Retraso		
Jitter		

Figura 5.22: Reconexión – Canal de Video

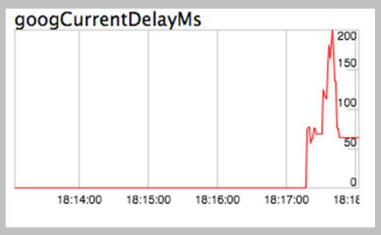
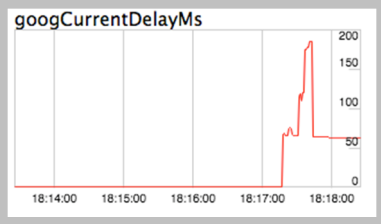
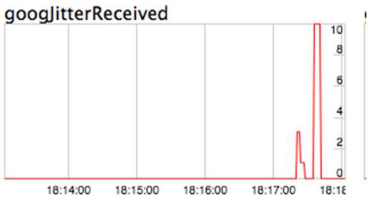
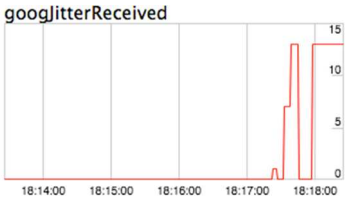
	Computador 1	Computador 2
Retraso		
Jitter		

Figura 5.23: Reconexión – Canal de Audio

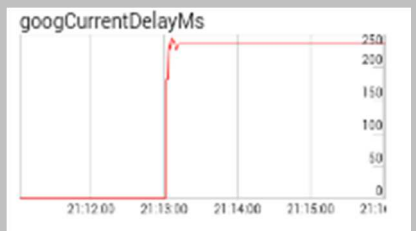
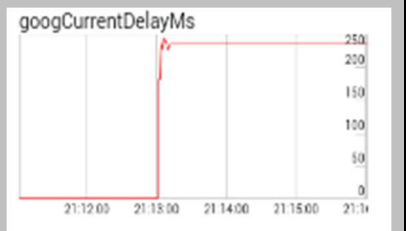
Escenario Dispositivo móvil – Dispositivo móvil

En la tabla 5.10 y la Figura 5.24-5.29 se muestran los resultados para una comunicación inicial, una interrupción y una reconexión. El análisis se realizó al canal de audio como al de video. De los resultados y aplicando la recomendación de la ITU-T G.114 [ITU, 2003] se concluye que el mecanismo de reconexión automática de la arquitectura ARW no afecta a la comunicación debido a que en ambos casos no superan los valores indicados en G.114 en jitter y Retraso.

La Figura 5.24 indica el retraso y jitter obtenidos en la comunicación inicial de la sesión para el canal de video. La Figura 5.25 indica el retraso y jitter obtenidos en la comunicación inicial de la sesión para el canal de audio. La Figura 5.26 indica el retraso y jitter obtenidos dada la interrupción de la sesión para el canal de video. La Figura 5.27 indica el retraso y jitter obtenidos dada la interrupción de la sesión para el canal de audio. La Figura 5.28 indica el retraso y jitter obtenidos dada la reconexión de la sesión para el canal de video. La Figura 5.29 indica el retraso y jitter obtenidos dada la reconexión de la sesión para el canal de audio.

Tabla 5.10: Resultados escenario Dispositivo móvil - Dispositivo móvil

ESCENARIO DISPOSITIVO MÓVIL – DISPOSITIVO M-OVIL		
	Dispositivo Móvil 1	Dispositivo Móvil 2
COMUNICACIÓN INICIAL		
Canal de Video		
Jitter	30ms	30ms
Retraso	250ms	250ms
Canal de Audio		
Jitter	20ms – 50ms	60ms – 20ms
Retraso	140ms – 10ms	100ms – 40ms
INTERRUPCIÓN		
Canal de Video		
Jitter	30ms	20ms – 60ms
Retraso	600ms – 500ms	700ms – 600ms
Canal de Audio		
Jitter	100ms – 30ms	60ms – 20ms
Retraso	250ms – 50ms	200ms – 50ms
RECONEXIÓN		
Canal de Video		
Jitter	40ms – 30ms	20ms – 60ms
Retraso	500ms	600ms – 500ms
Canal de Audio		
Jitter	40ms – 20ms	40ms – 20ms
Retraso	200ms – 50ms	200ms – 50ms

Video	Dispositivo Móvil 1	Dispositivo Móvil 2
Retraso		

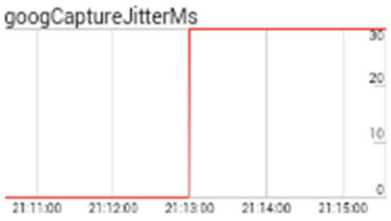
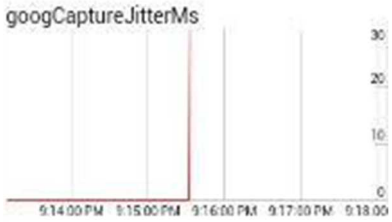
Jitter		

Figura 5.24: Comunicación inicial – Canal de Video

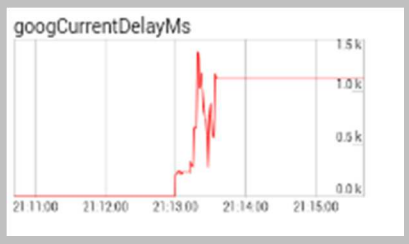
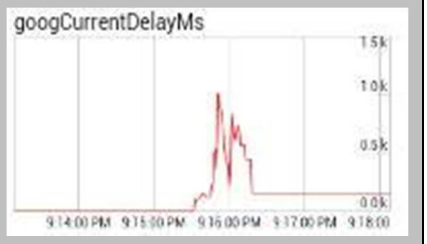
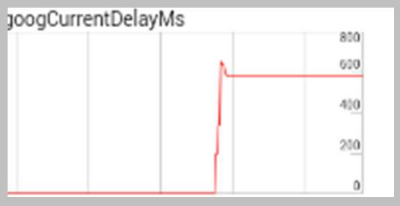
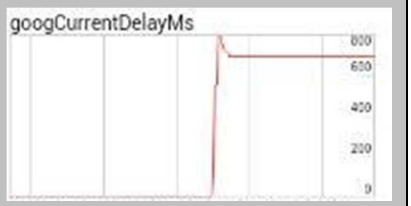
Audio	Dispositivo Móvil 1	Dispositivo Móvil 2
Retraso		
Jitter		

Figura 5.25: Comunicación inicial – Canal de Audio

	Dispositivo Móvil 1	Dispositivo Móvil 2
Retraso		

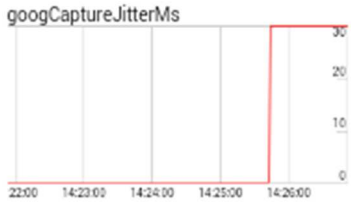
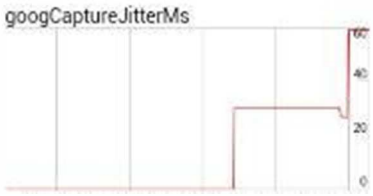
Jitter		

Figura 5.26: Interrupción – Canal de Video

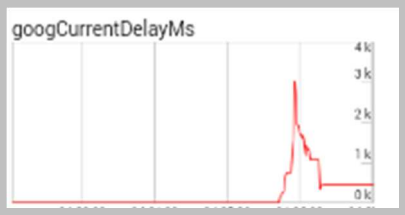
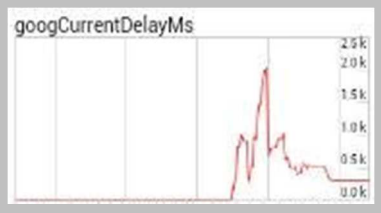
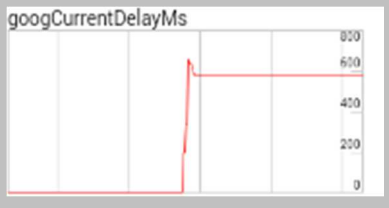
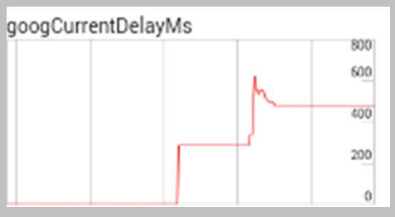
	Dispositivo Móvil 1	Dispositivo Móvil 2
Retraso		
Jitter		

Figura 5.27: Interrupción – Canal de Audio

	Dispositivo Móvil 1	Dispositivo Móvil 2
Retraso		

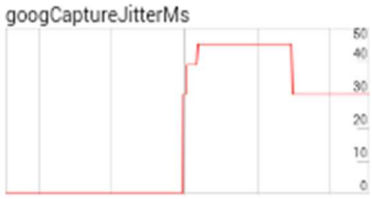
Jitter		

Figura 5.28: Reconexión – Canal de Video

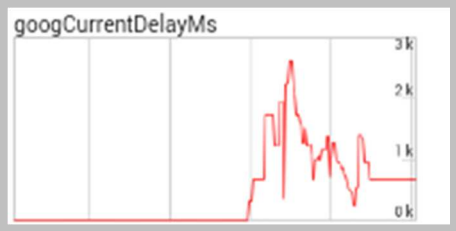
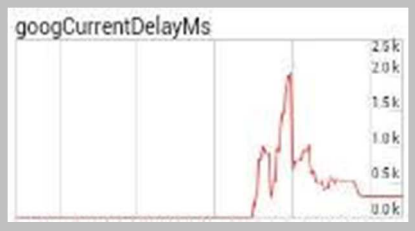
	Dispositivo Móvil 1	Dispositivo Móvil 2
Retraso		
Jitter		

Figura 5.29: Reconexión – Canal de Audio

Analizados ambos escenarios y los resultados obtenidos se determina que el mecanismo de reconexión automática de la arquitectura ARW no afecta la comunicación, sin embargo se podría analizar otros factores como el ancho de banda, intensidad de la señal, entre otros, para conocer como éstos están afectando la estabilidad de la comunicación.

Para la evaluación de la calidad del servicio se recomienda utilizar e interpretar otros parámetros que provee la herramienta chrome://webrct-internals, como el ancho de banda lo que podría apoyar a obtener mejores resultados y un mejor análisis de calidad de servicio ya que el jitter y el Retraso dependen de este parámetro.

COMPUTADOR – COMPUTADOR

Video

De acuerdo a los resultados que muestra la Tabla 5.9 (Resultados Escenario: Computador – Computador) se puede concluir que el módulo de reconexión no afecta a la comunicación de este canal ya que el jitter en ambos casos llega a un punto máximo 25ms mientras que el Retraso tiene una variación entre los 50 y 100ms para ambos casos y basados en la referencia para la calidad de servicio frente a estos valores la calidad es aceptable.

Audio

De acuerdo a los resultados que muestra la Tabla 5.9 (Resultados Escenario: Computador – Computador) se observa que el jitter tiene un máximo de 13ms y un Retraso que varía de entre los 105ms hasta los 200ms con conexión y cuando se produce una interrupción retoma los 200ms disminuyendo hacia los 55ms. De acuerdo a estos resultados y basados en la referencia para la calidad de servicio la calidad es aceptable.

DISPOSITIVO MÓVIL – DISPOSITIVO MÓVIL

Video

De acuerdo a los resultados que muestra la Tabla 5.10 (Resultados Escenario: Dispositivo Móvil – Dispositivo Móvil) se puede concluir que el módulo de reconexión no afecta a la comunicación de este canal ya que el jitter en ambos casos llega a un punto máximo 60ms lo que de acuerdo a la referencia es aceptable pero el Retraso sobrepasa los 400ms.

Audio

De acuerdo a los resultados que muestra la Tabla 5.10 (Resultados Escenario: Dispositivo Móvil – Dispositivo Móvil) se observa que el jitter alcanza su punto máximo a 100ms lo que se considera aceptable de acuerdo a la referencia. A su vez el Retraso que varía de

entre los 50ms y 200ms cuando hay conexión y varía entre el mismo valor cuando se produce una interrupción lo cual es aceptable.

Después del análisis de ambos escenarios y los resultados obtenidos se puede determinar que el módulo desarrollado no afecta la comunicación, sin embargo se podría analizar otros factores como el ancho de banda, intensidad de la señal, entre otros, para conocer como estos estar afectando la estabilidad de la comunicación.

Una de las grandes limitaciones de la arquitectura propuesta se encuentra en las limitaciones técnicas que presentan los dispositivos inalámbricos ante dispositivos de escritorio.

Ahora expones dos pruebas más realizadas en la Universidad de las Palmas de Gran Canaria. La primera experiencia se realizó en el pasillo del Pabellón C Sede del Departamento de Telemática y la segunda experiencia se la realizó entre el Laboratorio del Grupo de Arquitectura y Ocurrencia y la residencia de D. Álvaro Suárez Sarmiento.

Analicemos el primer caso, que corresponde al establecimiento del escenario de comunicación entre dos usuarios ubicados en el pasillo del Departamento de Telemática y al inicio de la sesión se presentaron los siguientes resultados en audio y video, indicados en las Figuras 5.30 y 5.31.

Podemos observar que al iniciar su establecimiento arranca con unos valores altos en jitter y delay, pero pasa el tiempo y empieza a mejorar. En relación a los paquetes perdidos observamos que mientras se establece la sesión no hay paquetes perdidos, posterior inicia la sesión empiezan a perderse paquetes pero muy pocos.

Posterior el usuario se desplazó en sentido de salir de cobertura y se presentan los resultados indicados en las Figuras 5.32 y 5.33.

Se observa que los parámetros de jitter y retraso empiezan a bajar y la pérdida de paquetes empieza a estabilizarse.

Finalmente se realizó la experiencia de reconexión obteniendo los resultados indicados en las Figuras 5.36 y 5.37.

Observamos que efectivamente estamos ingresando en una zona de interrupción y los parámetros de jitter y retraso empiezan a elevarse. Los paquetes perdidos empiezan a aumentar.

Ahora retornamos a una zona de cobertura y tenemos los resultados indicados en las Figuras 5.34 y 5.35.

Se observa que al retorno de la interrupción los parámetros se disparan como en la fase inicial y posterior se estabilizan.

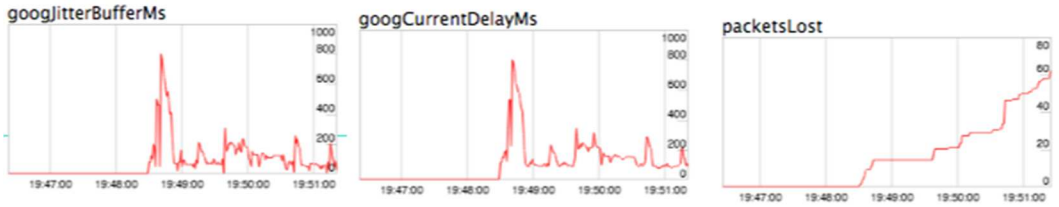


Figura 5.30: Resultados de canal de audio – fase inicial

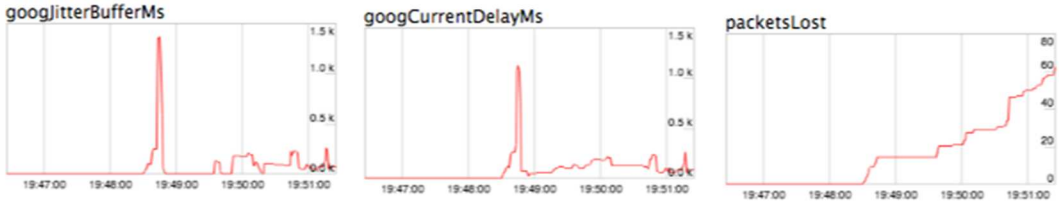


Figura 5.31: Resultado de canal de video – fase inicial

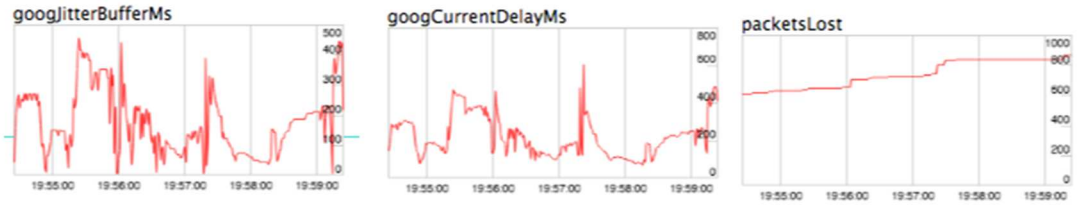


Figura 5.32: Resultados de canal de audio – zona de no cobertura

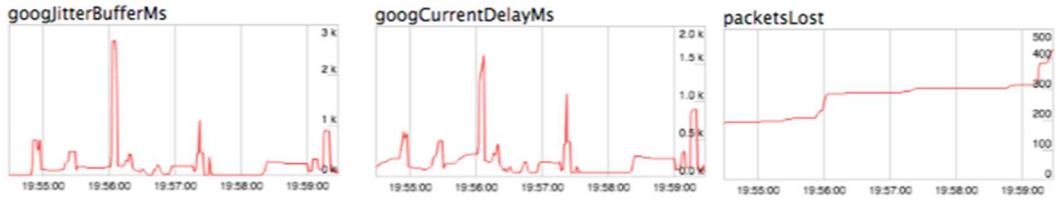


Figura 5.33: Resultados de canal de video – zona de no cobertura

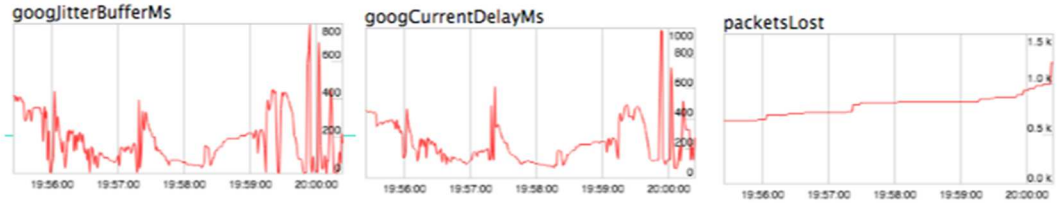


Figura 5.34: Resultados de canal de audio – zona de cobertura

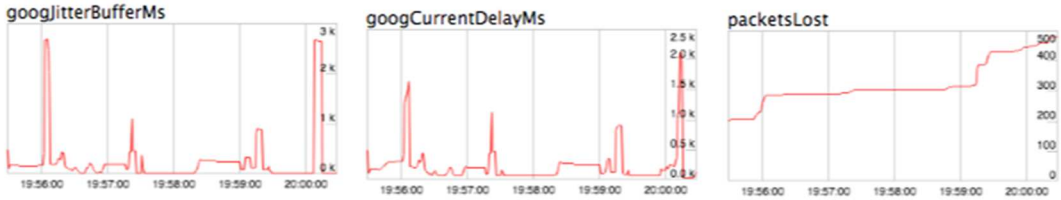


Figura 5.35: Resultados canal de video – zona de cobertura

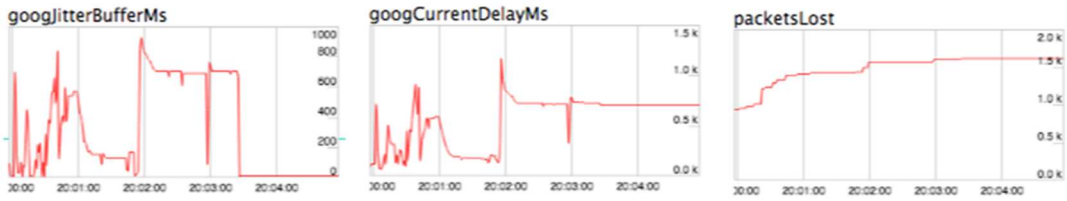


Figura 5.36: Resultados de canal de audio – retorno de una interrupción

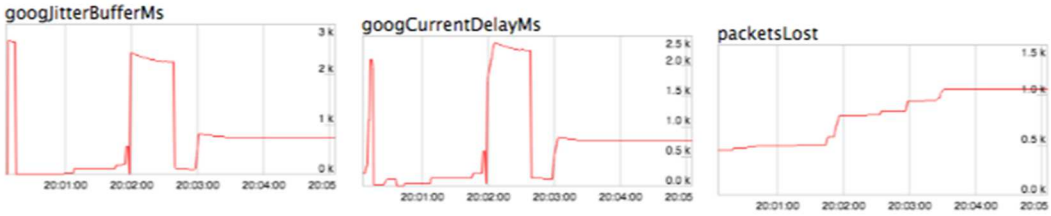


Figura 5.37: Resultado de canal de video – retorno de una interrupción

El segundo caso se la realizó entre el Laboratorio del Grupo de Arquitectura y Ocurrencia y la residencia de D. Álvaro Suárez Sarmiento.

La Figura 5.38 indica la sesión uno que presenta los escenarios dentro de zona de cobertura, proximidades de la zona de interrupción y retorno a zona de cobertura.

Las Figuras 5.39 y 5.40 presentan los resultados de inicio de sesión para los canales de audio y video.

Podemos observar que al iniciar su establecimiento arranca con unos valores altos en jitter y delay, pero pasa el tiempo y empieza a mejorar. En relación a los paquetes perdidos observamos que mientras se establece la sesión no hay paquetes perdidos, posterior inicia la sesión empiezan a perderse paquetes pero muy pocos.

Posterior el usuario se desplazó en sentido de salir de cobertura y se presentan los resultados indicados en las Figuras 5.41 y 5.42. Observamos que efectivamente estamos ingresando en una zona de interrupción y los parámetros de jitter y retraso empiezan a elevarse. Los paquetes perdidos empiezan a aumentar.

Ahora presentamos los resultados de una desconexión de sesión que se puede observar en las Figuras 5.43 y 5.44. Se observa que con los parámetros muy elevados, estos se estabilizan en cierto valor indicando que la desconexión se ha dado.

Finalmente tenemos el establecimiento de la nueva sesión y lo presentamos en la Figura 5.45. La Figura 5.46 presenta el resultado del canal de audio. Se observa que al retorno de la interrupción los parámetros se disparan como en la fase inicial y posterior se estabilizan.

Después del análisis de ambos escenarios y los resultados obtenidos se puede determinar que el módulo desarrollado no afecta la comunicación.

No afectan al tiempo de ejecución drásticamente la interrupción ni el nuevo establecimiento dado y el mecanismo de control de interrupción no genera incremento en el jitter y retraso de la sesión establecida.

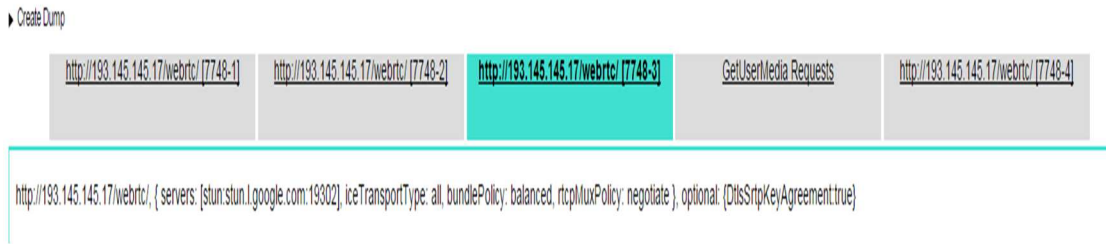


Figura 5.38: Sesión inicial de comunicación

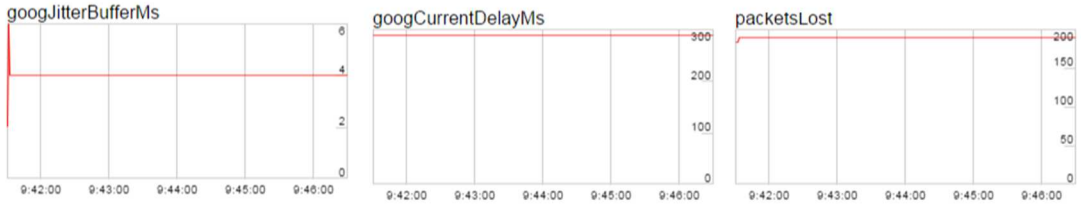


Figura 5.39: Resultados de inicio de sesión - canal de audio

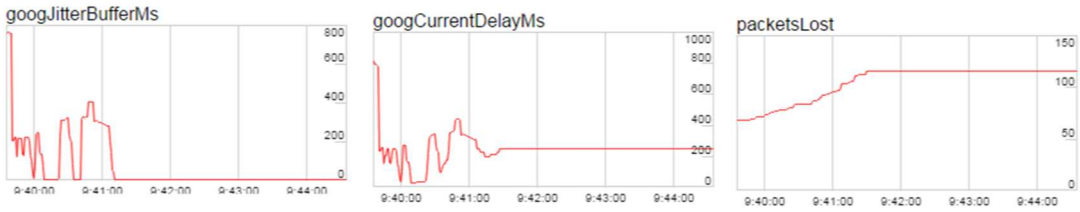


Figura 5.40: Resultados de inicio de sesión – canal de video

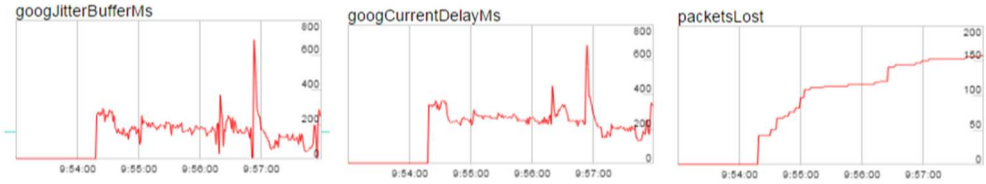


Figura 5.41: Resultados zona de no cobertura – canal de audio

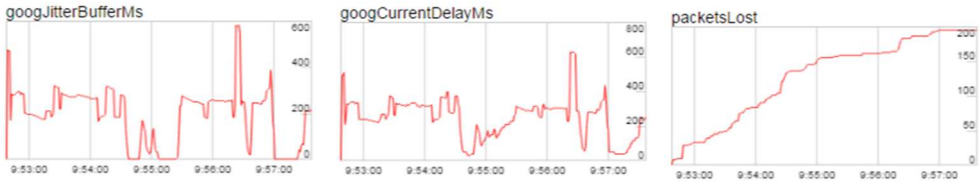


Figura 5.42: Resultados zona de no cobertura – canal de video

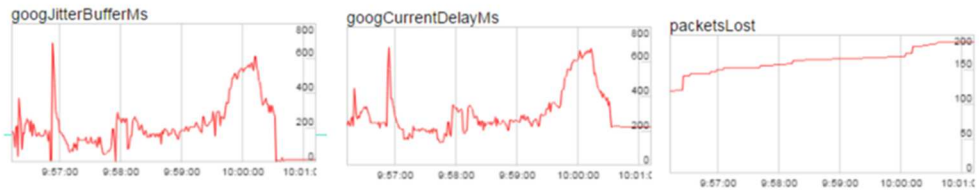


Figura 5.43: Resultados de una desconexión de sesión – canal de audio

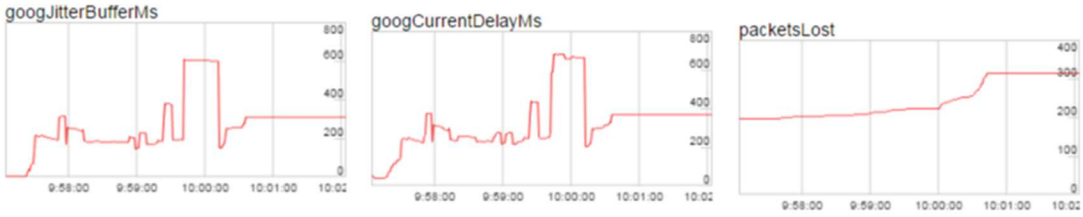


Figura 5.44: Resultados de una desconexión de sesión – canal de video

Figura 5.45: Nueva sesión reconectada

Figura 5.46: Resultados de la sesión reconectada – canal de audio.

Cabe señalar que en la revisión de la literatura no se hallaron referencias a estudios hechos que utilicen entrega de video streaming en tiempo real y que controle la interrupción. Esto hace que nuestra investigación sea pionera en el área y se diferencia de modo notable de otros enfoques propuestos por la comunidad científica.

6. Conclusiones y líneas de trabajo futuro

En este capítulo se presentan las principales conclusiones obtenidas en este trabajo de investigación y algunas líneas de posible trabajo futuro.

6.1. Conclusiones

Hemos abordado el problema de video streaming móvil mediante el diseño de un modelo basado en patrones software, que permita dar solución a problemas comunes en el desarrollo de software. El problema atacado es el incremento en el tiempo de ejecución del video, debido a las retransmisiones que producen los quiebres del canal de comunicación inalámbrica, y en consecuencia, los repetitivos tiempos de establecimiento de la sesión. Esto afecta negativamente la calidad de experiencia de usuario provocando malestar y hasta un posible abandono definitivo de la sesión. Se ha comprobado mediante el análisis de rendimiento del modelo matemático que el coste que genera las retransmisiones incrementa drásticamente el tiempo de ejecución del video streaming, generando congestión en la Red y minimizando acceso a recurso a otros usuarios y en consecuencia mala experiencia de usuario. Las soluciones que hemos aportado a este problema han sido el de diseñar un modelo base mediante patrones software que permita mediante incrementales de funcionalidad incorporar otros elementos que contribuyan a la mitigación del problema. Este modelo base fue revisado y publicado en [132] e implementado en algunos proyectos de investigación de la ESPE. Un segundo trabajo aportó con brindarle proactividad al usuario mediante publicación de alertas e información con RA en la pantalla ante posible ingreso a zona de interrupción, mejorando positivamente la calidad de experiencia de usuario. Finalmente dado que hoy en día tenemos el auge de la videoconferencia en la nube se implementó el mecanismo del modelo base en la plataforma WebRTC obteniendo excelentes resultados porque el usuario ya no tenía que preocuparse por restablecer su sesión y además brindarle continuidad del servicio; logrando buenos niveles de QoE que compensan totalmente el costo del almacenamiento de los videos mientras dura la interrupción.

Como fruto de este trabajo realizado se ha obtenido una experiencia personal de valor incalculable porque hemos aprendido a definir el estado del arte en base a una investigación bibliográfica y un análisis de profundo de artículos científicos, analizar desde el punto de vista del software soluciones para el campo de la telemática, proponer modelos basado en patrones software para el análisis de problemas de telemática, establecer escenarios de experimentación, levantar datos, analizar los resultados e

inferirlos, difunfir los resultados en base a artículos científicos técnicos, conocer el mundo de la investigación desde esta gran Universidad. Creemos que todo lo investigado y desarrollado puede tener aplicaciones en la educación, la medicina, la agricultura, debido a que nos permitiría dar soluciones a las interrupciones que presenta en canal inalámbrico al servicio de video streaming, que hoy en día es uno de los más utilizados por el mundo. En particular en nuestro trabajo personal va a aportar muchísimo porque nos hemos enriquecido del conocimiento para poder compartirlo con nuestros estudiantes de la Universidad de las Fuerza Armadas ESPE y contribuir en la generación de soluciones a los problemas que tiene la sociedad en este campo.

Creemos que hemos contribuido a la industria del video streaming porque hemos planteado soluciones que mejoran este servicio considerablemente en los tiempos de ejecución del video, evitando la retransmisión de flujos que congestionan la Red y elevando considerablemente la calidad de experiencia de usuario. Esto ha permitido incursionar en alternativas como es la videoconferencia en la nube para probar que nuestro modelo base funciona muy satisfactoriamente, teniendo como resultado una excelente percepción por el usuario. Además creemos que se generó un patrón software que permite dar solución a la interrupción del servicio de video streaming por una disrupción del canal de comunicación inalámbrica.

6.2. Líneas de trabajo futuro

Fruto de nuestro trabajo se abren nuevas líneas de trabajo futuro:

- Utilizar Peer to Peer Straightforward Protocolo que es un protocolo para el video streaming de contenido multimedia sobre internet, para implantarlo con el modelo para video en tiempo real y evaluar el comportamiento ante interrupciones tanto en escenarios de secuencias en directo como en difusión de secuencias ya almacenadas.
- Probar cómo va el modelo base diseñado en aplicaciones que realizan streaming de vídeo, ya que se está apostando con fuerza por el

periodismo ciudadano y que trata de acercarnos a una versión más humanizada o empática de la información.

- Hasta el momento se ha probado en un ambiente servidor – dispositivo móvil, ahora sería interesante probar el mecanismo de interrupción como funciona si ahora el dispositivo inalámbrico emite vídeo en directo a todos los dispositivos inalámbricos que se encuentren conectados.
- Sería interesante desplegar este mecanismo en sistemas de seguridad, donde ahora el video streaming sería la alternativa para despliegue de información de algún suceso negativos, debido a que el mecanismo almacenaría los flujos de video capturados por el quiebre del canal inalámbrico y estarían en espera de ser transmitidos a algún lugar del mundo.

Glosario

3GPP	3rd Generation Partnership Project
ACL	Agent Communications Language
ADSL	Asymmetric Digital Subscriber Line
AMS	Agent Management System
APC	Agent Proxy Client
API	Application Programming Interface
APS	Agent Proxy Server
BRMS	Business Rule Management System
CDN	Content Delivery Network
CSS	Cascading Style Sheets
CODEC	Compresor Descompresor
DASH	Dynamic Adaptive Streaming over HTTP
DF	Directory Facilitator
FIPA	Foundation for Intelligent Physical Agent
Ginga-NCL	Ginga Nested Context Language
GPS	Global Positioning system
HMD	Head Mounted Display
HomeRF	Home Radio Frequency
HTTP	Hypertext Transfer Protocol
ICE	Interactive Connection Establishment
IDC	International Data Corporation
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IPTV	Internet Proctocol Television
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union

JADE	Java Agent Development Framework
JSEP	Java Session Establishment Protocol
KML	Keyhole Markup Language
KMZ	Keyhole Markup Language Zip
LEAP	Lightweight Extensible Agent Platform
LGPL	Lesser General Public License
LiDAR	Laser Imaging Detection and Ranging
LTE	Long Term Evolution
MANET	Mobile Ad Hoc Networks
MMS	Microsoft Media Server
MOS	Mean Opinion Score
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
MPQM	Moving Pictures Quality Metric
MTP	Message Transport Protocol
MVC	Model View Controller
NAT	Network Address Translation
NQM	Structural Similarity Index
P2P	Peer-to-Peer
PCMP	Persistent Connectivity Management Protocol
PDA	Personal Digital Assistant
PSNR	Peak-Signal-to-Noise-Ratio
QoE	Quality of Experience
QoS	Quality of Service
RMI	Remote Method Invocation
RP	Remote Programming
RPC	Remoto Procedure Control
RSSI	Received Signal Strength Indicator
RTC	Real Time Communication
RTCP	Real Time Control Protocol
RTMFP	Real Time Media Flow Protocol

RTMP	Real Time Messaging Protocol
RTP	Real Time Transport Protocol
RTSP	Real Time Streaming Protocol
SDP	Session Description Protocol
SRTP	Secure Real-Time Transport Protocol
SS	Signal Strength
SSIM	Structural Similarity Index
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TURN	Traversal Using Relay NAT
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VoD	Video on Demand
VoIP	Voice Over Internet Protocol
VQM	Video Quality Metric
W3C	World Wide Web Consortium
WebRTC	Web Real-Time Communication
WiFi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WQMPT	Worldwide Quarterly Mobile Phone Tracker
XML	eXtensible Markup Language

Referencias bibliográficas

[1] W.-F. Alliance, "Wi-fi alliance," *available at: www. wi-fi. org*, 2009.

[2] K. Leung, "WiMAX forum/3GPP2 proxy mobile IPv4," 2010.

[3] G. Roth, R. Sjöberg, G. Liebl, T. Stockhammer, V. Varsa, and M. Karczewicz, "Common test conditions for RTP/IP over 3GPP/3GPP2," *ITU-T SG16 Doc. VCEG-M77, Austin, TX, USA*, 2001.

[4] H. Holma and A. Toskala, *HSDPA/HSUPA for UMTS: high speed radio access for mobile communications*. John Wiley & Sons, 2007.

[5] S. Sesia, I. Toufik, and M. Baker, *LTE: the UMTS long term evolution*. Wiley Online Library, 2009.

[6] M. Papadopouli and H. Schulzrinne, "Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 2001, pp. 117–127.

[7] S. Babin, *Developing Software for Symbian OS 2nd Edition: A Beginner's Guide to Creating Symbian OS v9 Smartphone Applications in C++*. John Wiley & Sons, 2008.

[8] H. Schulzrinne, A. Rao, and R. Lanphier, "RFC 2326: real time streaming protocol," *Availab le on http://www. ietf. org/rfc/rfc2326. txt*, 1998.

[9] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "JADE: A software framework for developing multi-agent applications. Lessons learned," *Information and Software Technology*, vol. 50, no. 1, pp. 10–21, 2008.

[10] G. Caire and F. Pieri, "Leap user guide," *TILab, Jan*, 2006.

[11] X. Qiu, H. Liu, D. Li, S. Zhang, D. Ghosal, and B. Mukherjee, "Optimizing HTTP-based Adaptive Video Streaming for wireless access networks," in *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, 2010, pp. 838–845.

[12] D. Bulira and K. Walkowiak, "Voice and video streaming in wireless computer networks-evaluation of network delays," in *Future Internet Communications (BCFIC), 2012 2nd Baltic Congress on*, 2012, pp. 156–161.

[13] S. U. Rehman, T. Turletti, and W. Dabbous, "Multicast video streaming over WiFi networks: Impact of multipath fading and interference," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, 2011, pp. 37–42.

[14] A. G. Fragkiadakis, E. Z. Tragos, and I. G. Askoxylakis, "Video streaming performance in wireless hostile environments," in *Multimedia and Ubiquitous Engineering (MUE), 2011 5th FTRA International Conference on*, 2011, pp. 267–272.

[15] T. Liu and C. Choudary, "Content-aware streaming of lecture videos over wireless networks," in *Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on*, 2004, pp. 458–465.

[16] J. Xiao, T. Tillo, C. Lin, Y. Zhang, and Y. Zhao, "A real-time error resilient video streaming scheme exploiting the late-and early-arrival packets," *Broadcasting, IEEE Transactions on*, vol. 59, no. 3, pp. 432–444, 2013.

[17] J. P. Sterbenz, D. Hutchison, E. K. Cetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Redundancy, diversity, and connectivity to achieve multilevel network resilience, survivability, and disruption tolerance invited paper," *Telecommunication Systems*, vol. 56, no. 1, pp. 17–31, 2014.

[18] D. H. Bui, K. Lee, S. Oh, I. Shin, H. Shin, H. Woo, and D. Ban, "Greenbag: Energy-efficient bandwidth aggregation for real-time streaming in heterogeneous mobile wireless networks," in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, 2013, pp. 57–67.

[19] O. Oyman, J. Foerster, Y. Tcha, and S.-C. Lee, "Toward enhanced mobile video services over WiMAX and LTE [WiMAX/LTE update]," *Communications Magazine, IEEE*, vol. 48, no. 8, pp. 68–76, 2010.

[20] M. Gorius, Y. Shuai, and T. Herfet, "Dynamic media streaming over wireless and mobile ip networks," in *Consumer Electronics-Berlin (ICCE-Berlin), 2012 IEEE International Conference on*, 2012, pp. 158–162.

[21] A. A. Rahman, Y. A. Syahbana, K. A. Bakar, and others, "Nonlinearity modelling of QoE for video streaming over wireless and mobile network," in *Intelligent Systems, Modelling and Simulation (ISMS), 2011 Second International Conference on*, 2011, pp. 313–317.

[22] S. Jumisko-Pyykkö, J. Häkkinen, and G. Nyman, "Experienced quality factors: qualitative evaluation approach to audiovisual quality," in *Electronic Imaging 2007*, 2007, p. 65070M–65070M.

[23] A. Seetharam, P. Dutta, V. Arya, J. Kurose, M. Chetlur, and S. Kalyanaraman, "On managing quality of experience of multiple video streams in wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 14, no. 3, pp. 619–631, 2015.

[24] I. Cotanis, "Performance evaluation of objective QoE models for mobile voice and video-audio services," in *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*, 2013, pp. 452–457.

[25] J. De Vriendt, D. De Vleeschauwer, and D. C. Robinson, "QoE model for video delivered over an LTE network using HTTP adaptive streaming," *Bell Labs Technical Journal*, vol. 18, no. 4, pp. 45–62, 2014.

[26] J. Nightingale, Q. Wang, C. Grecos, and S. Goma, "The impact of network impairment on quality of experience (QoE) in H. 265/HEVC video streaming," *Consumer Electronics, IEEE Transactions on*, vol. 60, no. 2, pp. 242–250, 2014.

[27] Y. Wang, "Survey of objective video quality measurements," 2006.

[28] S. Winkler and P. Mohandas, "The evolution of video quality measurement: from PSNR to hybrid metrics," *Broadcasting, IEEE Transactions on*, vol. 54, no. 3, pp. 660–668, 2008.

[29] B. Lewcio and S. Möller, "A testbed for QoE-based multimedia streaming optimization in heterogeneous wireless networks," in *Signal Processing and Communication Systems (ICSPCS), 2011 5th International Conference on*, 2011, pp. 1–9.

[30] O. Ognenoski, M. Razaak, M. G. Martini, and P. Amon, "Medical video streaming utilizing MPEG-DASH," in *e-Health Networking, Applications & Services (Healthcom), 2013 IEEE 15th International Conference on*, 2013, pp. 54–59.

[31] G. Caire, G. Lavarone, and others, "Jade Tutorial, Jade Programming for Android," *2012-06-14). http://www. doc-txt. com/JADE-ANDROID-Guide. pdf*. 2011.

[32] D. Gotta, T. Trucco, M. Ughetti, S. Semeria, C. Cucè, and A. M. Porcino, "JADE Android Add-on Guide." TILAB, 2008.

[33] Y. D. González and Y. F. Romero, "Patrón Modelo-Vista-Controlador.," *Revista Telem@ tica*, vol. 11, no. 1, pp. 47–57, 2012.

[34] "The Path towards Gb/s Wireless LANs." .

[35] E. Perahia and M. X. Gong, "Gigabit wireless LANs: an overview of IEEE 802.11 ac and 802.11 ad," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 15, no. 3, pp. 23–33, 2011.

[36] L. Verma, M. Fakharzadeh, and S. Choi, "Wifi on steroids: 802.11 ac and 802.11 ad," *Wireless Communications, IEEE*, vol. 20, no. 6, pp. 30–35, 2013.

[37] R.-S. Cheng, "Performance evaluation of stream control transport protocol over IEEE 802.11 ac networks," in *Wireless Communications and Networking Conference Workshops (WCNCW), 2015 IEEE*, 2015, pp. 97–102.

[38] S. Gadhiya, K. Wandra, V. B. Vaghela, and others, "Role of mobile augmentation in mobile application development," in *Engineering Education: Innovative Practices and Future Trends (AICERA), 2012 IEEE International Conference on*, 2012, pp. 1–5.

[39] J. McMullan and I. Richardson, “The mobile phone: a hybrid multi-platform medium,” in *Proceedings of the 3rd Australasian conference on Interactive entertainment*, 2006, pp. 103–108.

[40] T. Chen, Q. Shi, X. Lou, and W. Hu, “A case study of course design for software development on mobile phone,” in *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on*, 2010, pp. 59–64.

[41] W. You, K. Qian, D. C.-T. Lo, P. Bhattacharya, W. Chen, T. Rogers, J.-C. Chern, and J. Yao, “Promoting mobile computing and security learning using mobile devices,” in *Integrated STEM Education Conference (ISEC), 2015 IEEE*, 2015, pp. 205–209.

[42] C. System and Inc, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014-2019.” .

[43] J. Gruskovnjak and A. L. y Stuart Taylor, “El Nuevo Mundo de Wi-Fi para los Proveedores de Servicios.” .

[44] S. Juan and T. Shoulian, “Operator’s Mobile Internet Strategy in the Process of Converged Network,” in *Management and Service Science (MASS), 2010 International Conference on*, 2010, pp. 1–4.

[45] N. Kara and V. Planat, “Performance analysis of IP multimedia services over HSDPA mobile networks,” in *IP Multimedia Subsystem Architecture and Applications, 2007 International Conference on*, 2007, pp. 1–5.

[46] C.-H. Fu, Y.-L. Chan, T.-P. Ip, and W.-C. Siu, “New architecture for MPEG video streaming system with backward playback support,” *Image Processing, IEEE Transactions on*, vol. 16, no. 9, pp. 2169–2183, 2007.

[47] R. Ferzli, M. P. McGarry, and M. A. Al-Alaoui, “Effect of hand jitter on video compression algorithms for mobile devices,” in *Communications and Information Technology (ICCIT), 2013 Third International Conference on*, 2013, pp. 330–334.

[48] Y. S. Baguda, N. Fisal, S. H. Syed, S. K. Yusof, and R. Rashid, “H264/AVC features & functionalities suitable for wireless video transmission,” in *Wireless and Optical Communications Networks, 2008. WOCN’08. 5th IFIP International Conference on*, 2008, pp. 1–5.

[49] K. K. Kambhatla, S. Kumar, S. Paluri, and P. C. Cosman, “Wireless H. 264 video quality enhancement through optimal prioritized packet fragmentation,” *Multimedia, IEEE Transactions on*, vol. 14, no. 5, pp. 1480–1495, 2012.

[50] Y. Huang and S. Mao, “Downlink power control for multi-user VBR video streaming in cellular networks,” *Multimedia, IEEE Transactions on*, vol. 15, no. 8, pp. 2137–2148, 2013.

[51] A. Luthra, S. Wenger, W. Zhu, and others, "Introduction to the special issue on streaming video," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 3, pp. 265–268, 2001.

[52] A. Raghuveer, E. Kusmierek, and D. H. Du, "A network-aware approach for video and metadata streaming," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 8, pp. 1028–1040, 2007.

[53] Y. Liu and M. Hefeeda, "Video streaming over cooperative wireless networks," in *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, 2010, pp. 99–110.

[54] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt, "Micro-blog: sharing and querying content through mobile phones and social participation," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, 2008, pp. 174–186.

[55] P. Shankar, Y.-W. Huang, P. Castro, B. Nath, and L. Iftode, "Crowds replace experts: Building better location-based services using mobile social network interactions," in *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, 2012, pp. 20–29.

[56] Z. Qin, D.-J. Li, and M. C. Chuah, "Lehigh Explorer: A Real time video streaming application with mobility support for content centric networks," in *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, 2013, pp. 33–40.

[57] N. Ravi, T. Mala, M. K. Srinivasan, and K. Sarukesi, "Design and Implementation of VOD (Video on Demand) SaaS Framework for Android Platform on Cloud Environment," in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, 2013, vol. 2, pp. 171–176.

[58] G. O'Driscoll, *Next generation IPTV services and technologies*. John Wiley & Sons, 2008.

[59] L. Tseng, H.-C. Chuang, C. Y. Huang, and T. Chiang, "A buffer-feedback rate control method for video streaming over mobile communication systems," in *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, 2005, vol. 2, pp. 1266–1270.

[60] S. Mack, *Streaming media bible*. John Wiley & Sons, Inc., 2002.

[61] M. Al-Hami, A. Khreishah, and J. Wu, "Video Streaming Over Wireless LAN With Network Coding," in *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*, 2013, pp. 173–176.

[62] A. K. S. H. S. Dashti, R. Zimmermann, and S. H. Kim, *Streaming media server design*. Prentice Hall Professional Technical Reference, 2003.

[63] B. Girod, J. Chakareski, M. Kalman, Y. Liang, E. Setton, and R. Zhang, "Advances in network-adaptive video streaming," in *2002 Tyrrhenian Inter. Workshop on Digital Communications*, 2002.

[64] L. Gao, Z.-L. Zhang, and D. Towsley, "Proxy-assisted techniques for delivering continuous multimedia streams," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 6, pp. 884–894, 2003.

[65] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1999, vol. 3, pp. 1310–1319.

[66] E. Ohwovoriole and Y. Andreopoulos, "Rate-distortion performance of contemporary video codecs: Comparison of Google/WebM VP8, AVC/H. 264, and HEVC TMuC," in *Proc. London Communications Symposium (LCS)*, 2010, pp. 1–4.

[67] D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe, "Multimedia storage servers: A tutorial," *Computer*, no. 5, pp. 40–49, 1995.

[68] H. Schulzrinne, "Real time streaming protocol (RTSP)," 1998.

[69] J. Postel, "RFC793: Transmission Control Protocol, Sept. 1981," *URL http://www. rfc. net/rfc793. txt. See also STD0007. Status: STANDARD*, 1997.

[70] J. Postel, "RFC 768: User datagram protocol, 1980," *URL http://www. ietf. org/rfc/rfc768. txt*, 1980.

[71] H. S. RFC3550, S. CASNER, R. FREDERICK, and V. JACOBSON, "RTP: A Transport Protocol for Rea-l Time Applications." S. l.]: RFC Editor United States, 2003.

[72] C. HU1TEMA, "Rfc3605," *Real time control protocol (RTCP) attribute in session description protocol (SDP)*, 2003.

[73] M. H. U. V. J. Packet, "Design C Perkins University of Glasgow:" SDP: Session Description Protocol; rfc4566. txt," *IETF Standard, Internet Engineering Task Force, IETF, CH*, pp. 0000–0003, 2006.

[74] L. F. Ludwig, J. C. Lauwers, K. A. Lantz, G. J. Burnett, and E. R. Burns, "Participant display and selection in video conference calls." Google Patents, 2007.

[75] D. Braun, "Method and means for interactive audio and video conferencing." Google Patents, 1982.

[76] L. L. Chang, "Method and system for participating locations in a multi-point video conference." Google Patents, 2005.

[77] L. Abeni, G. Lipari, and G. Buttazzo, "Constant bandwidth vs. proportional share resource allocation," in *Multimedia Computing and Systems, 1999. IEEE International Conference on*, 1999, vol. 2, pp. 107–111.

[78] Z. Deng, X. Zhang, and D. Yang, "Pragmatic quality of experience optimization for wireless multimedia applications on intelligent terminals," in *Global High Tech Congress on Electronics (GHTCE), 2013 IEEE*, 2013, pp. 214–218.

[79] V. Ramamurthi, O. Oyman, and J. Foerster, "Video-QoE aware resource management at network core," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, 2014, pp. 1418–1423.

[80] Y. A. P. Leal and L. G. M. Ballesteros, "Factores que Influyen en la Evaluación de QoE del Servicio de Video Streaming."

[81] T. Räty, J. Oikarinen, and M. Sihvonen, "A scalable quality of service middleware system with passive monitoring agents over wireless video transmission," in *Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the*, 2007, pp. 123–130.

[82] K. Seshadrinathan and A. C. Bovik, "Motion-based perceptual quality assessment of video," in *IS&T/SPIE Electronic Imaging*, 2009, p. 72400X–72400X.

[83] G. J. Sullivan and T. Wiegand, "Video compression-from concepts to the H. 264/AVC standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, 2005.

[84] J.-N. Hwang, *Multimedia networking: from theory to practice*. Cambridge University Press, 2009.

[85] A. Ksentini, M. Naimi, and A. Guéroui, "Toward an improvement of H. 264 video transmission over IEEE 802.11 e through a cross-layer architecture," *Communications Magazine, IEEE*, vol. 44, no. 1, pp. 107–114, 2006.

[86] K. Atalah, "Contributions to effective protocol design to mitigate continuous multimedia services short disriptions in WiFi networks," 2011.

[87] R. J. M. Tejedor, "WebRTC (Web Real-Time Communications)," *Bit*, no. 197, p. 18, 2014.

[88] M.-C. Hsu and Y.-C. Chen, "Enhanced pcf protocols for real-time multimedia services over 802.11 wireless networks," in *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on*, 2006, pp. 56–56.

[89] M. van der Schaar and P. A. Chou, *Multimedia over IP and wireless networks: compression, networking, and systems*. Academic Press, 2011.

[90] T. S. Rappaport and others, *Wireless communications: principles and practice*, vol. 2. prentice hall PTR New Jersey, 1996.

[91] J. Postel and others, "RFC 791: Internet protocol," 1981.

[92] H. Wang, L. P. Kondi, A. Luthra, and S. Ci, "Wireless Video Streaming," *4G Wireless Video Communications*, pp. 369–387.

[93] M. El-Gendy, A. Bose, K. G. Shin, and others, "Evolution of the Internet QoS and support for soft real-time applications," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1086–1104, 2003.

[94] H. Zhai, X. Chen, and Y. Fang, "A call admission and rate control scheme for multimedia support over IEEE 802.11 wireless LANs," *Wireless Networks*, vol. 12, no. 4, pp. 451–463, 2006.

[95] R. El-Marakby and M. Enugula, "Enhanced QoS for Real-time Multimedia Delivery over the Wireless Link using RFID Technology," in *Signal Processing and Information Technology, 2006 IEEE International Symposium on*, 2006, pp. 728–734.

[96] N. Cranley and M. Davis, "An experimental investigation of IEEE 802.11 e TXOP facility for real-time video streaming," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, 2007, pp. 2075–2080.

[97] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the Internet: approaches and directions," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 3, pp. 282–300, 2001.

[98] P. Prabhavathy and S. Bose, "Path stream group level encoding: Efficient wireless XML streaming," in *Recent Trends in Information Technology (ICRTIT), 2013 International Conference on*, 2013, pp. 582–589.

[99] N. V. Uti and R. Fox, "Testing the Computational Capabilities of Mobile Device Processors: Some Interesting Benchmark Results," in *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on*, 2010, pp. 477–481.

[100] M. R. Zakerinasab and M. Wang, "A cloud-assisted energy-efficient video streaming system for smartphones," in *Quality of Service (IWQoS), 2013 IEEE/ACM 21st International Symposium on*, 2013, pp. 1–10.

[101] V. Garg, *Wireless Communications & Networking*. Morgan Kaufmann, 2010.

[102] P. Hıpola and B. Vargas-Quesada, "Agentes inteligentes: definición y tipologıa: los agentes de la información," *El profesional de la información*, vol. 8, no. 4, pp. 13–21, 1999.

[103] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 02, pp. 115–152, 1995.

[104] R. Jensen, "Agent management system." Google Patents, 2007.

[105] W. Brenner, R. Zarnekow, and H. Wittig, *Intelligent software agents: foundations and applications*. Springer Science & Business Media, 2012.

[106] V. A. Pham and A. Karmouch, "Mobile software agents: an overview," *Communications Magazine, IEEE*, vol. 36, no. 7, pp. 26–37, 1998.

[107] F. Bellifemine, G. Caire, G. Rimassa, A. Poggi, T. Trucco, E. Cortese, F. Quarta, G. Vitaglione, N. Lhuillier, and J. Picault, "Java Agent Development Framework," *TILAB Italia, http://jade. cselt. it, status*, vol. 10, p. 2002, 2002.

[108] A. Pokahr, L. Braubach, and A. Walczak, "Jadex user guide," *Hamburg, Germany*, 2007.

[109] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE–A FIPA-compliant agent framework," in *Proceedings of PAAM*, 1999, vol. 99, no. 97–108, p. 33.

[110] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*, vol. 7. John Wiley & Sons, 2007.

[111] A. Moreno, A. Valls, and A. Viejo, *Using JADE-LEAP implement agents in mobile devices*. Universitat Rovira i Virgili. Departament d'Enginyeria Informàtica, 2003.

[112] Y. Wang, Z.-L. Zhang, D. H. Du, and D. Su, "A network-conscious approach to end-to-end video delivery over wide area networks using proxy servers," in *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1998, vol. 2, pp. 660–667.

[113] S. Cha, W. Du, and B. J. Kurz, "Middleware framework for disconnection tolerant mobile application services," in *Communication Networks and Services Research Conference (CNSR), 2010 Eighth Annual*, 2010, pp. 334–340.

[114] Y.-C. Lee and S.-H. Park, "RSSI-based fingerprint map building for indoor localization," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2013 10th International Conference on*, 2013, pp. 292–293.

[115] J. Chakareski, P. Chou, and others, "RaDiO Edge: Rate-distortion optimized proxy-driven streaming from the network edge," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 6, pp. 1302–1312, 2006.

[116] P. Bellavista and A. Corradi, "A QoS management middleware based on mobility prediction for multimedia service continuity in the wireless internet," in *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, 2004, vol. 1, pp. 531–538.

[117] P. Bellavista, A. Corradi, and C. Giannelli, "Mobile proxies for proactive buffering in wireless internet multimedia streaming," in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, 2005, pp. 297–304.

[118] Y.-S. Chen, T.-L. Chin, and Y.-C. Huang, "Collaborative localization in Wireless Sensor Networks based on dependable RSSI," in *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*, 2012, pp. 341–347.

[119] P. Bellavista and A. Corradi, "How to support Internet-based distribution of video on demand to portable devices," in *Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on*, 2002, pp. 126–132.

[120] S. Chan, G. Sohn, L. Wang, and W. Lee, "Dynamic WIFI-Based Indoor Positioning in 3D Virtual World," *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, no. 4, pp. 1–6, 2013.

[121] D.-H. Nam and S.-K. Park, "Adaptive multimedia stream presentation in mobile computing environment," in *TENCON 99. Proceedings of the IEEE Region 10 Conference*, 1999, vol. 2, pp. 966–969.

[122] P. Bellavista, A. Corradi, and C. Giannelli, "Mobile proxies for proactive buffering in wireless internet multimedia streaming," in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, 2005, pp. 297–304.

[123] P. Bellavista, A. Corradi, and L. Foschini, "Java-based proactive buffering for multimedia streaming continuity in the wireless Internet," in *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, 2005, pp. 448–450.

[124] J. Ott and D. Kutscher, "A disconnection-tolerant transport for drive-thru internet environments," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005, vol. 3, pp. 1849–1862.

[125] K. D. Teket, M. Sayit, and G. Kardas, "Software agents for peer-to-peer video streaming," *IET Software*, vol. 8, no. 4, pp. 184–192, 2014.

[126] C. E. Palazzi and A. Bujari, "A delay/disruption tolerant solution for mobile-to-mobile file sharing," in *Wireless Days (WD), 2010 IFIP*, 2010, pp. 1–5.

[127] U. Kumar and S. Agarwal, "Coding to Mitigate Video Disruption during Wireless Access Switching," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1–5.

[128] D. Benferhat, F. Guidec, and P. Quinton, "Disruption-tolerant wireless biomedical monitoring for marathon runners: A feasibility study," in *Wireless Personal Multimedia Communications (WPMC), 2011 14th International Symposium on*, 2011, pp. 1–5.

[129] A. Suarez, E. Macıas, J. Martın, Y. Gutiérrez, and M. Gil, "Light Protocol and Buffer Management for Automatically Recovering Streaming Sessions in WiFi Mobile Telephones," in *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM'08. The Second International Conference on*, 2008, pp. 70–76.

[130] E. Macias and A. Suarez, "Proactive estimation of the video streaming reception quality in WiFi networks using a cross-layer technique," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 7, no. 3, pp. 383–389, 2009.

[131] A. Suarez Sarmiento, F. Espino, and E. Macias, "Automatic recovering of RTSP sessions in mobile telephones using JADE-LEAP," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 7, no. 3, pp. 410–417, 2009.

[132] T. Gualotuña, D. Marcillo, E. M. López, and A. Suárez-Sarmiento, "Mobile Video Service Disruptions Control in Android Using JADE," in *Advances in Computing and Communications*, Springer, 2011, pp. 481–490.

[133] A. Suárez, M. La-Menza, E. M. Macıas, and V. S. Sunderam, "Automatic Resumption of Streaming Sessions over Wireless Communications Using Agents.," in *IMECS*, 2006, pp. 926–931.

[134] A. Suárez, M. La-Menza, E. M. Macıas, and V. Sunderam, "Automatic resumption of streaming sessions over WiFi using JADE," *IAENG International Journal of Computer Science*, vol. 33, no. 1, pp. 92–100, 2007.

[135] T. O'reilly, "What is Web 2.0: Design patterns and business models for the next generation of software," *Communications & strategies*, no. 1, p. 17, 2007.

[136] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[137] D. Van Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," *International Journal of Virtual Reality*, vol. 9, no. 2, p. 1, 2010.

[138] J. Fombona Cadavieco, M. Á. Pascual Sevillano, and M. F. Madeira Ferreira Amador, "Realidad aumentada, una evolución de las aplicaciones de los dispositivos móviles," 2012.

[139] S. Bellón Alcarazo, J. Creixel Rojo, and Á. Serrano Laguna, "Look!: framework para aplicaciones de realidad aumentada en Android," 2011.

[140] P. Browne, *JBoss Drools business rules*. Packt Publishing Ltd, 2009.

[141] D. Nolan and D. T. Lang, "Keyhole markup language," in *XML and Web Technologies for Data Sciences with R*, Springer, 2014, pp. 581–618.

[142] Y. Du, C. Yu, and L. Jie, "A study of GIS development based on KML and Google Earth," in *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, 2009, pp. 1581–1585.

[143] O. G. Consortium and others, "OGC KML," *Open Geospatial Consortium (OGC), Standard OGC*, 2009.

[144] J. Rohlf and B. McClendon, “Markup language for an interactive geographic information system.” Google Patents, 2008.

[145] M. Pilgrim, *HTML5: up and running*. O"Reilly Media, Inc., 2010.

[146] A. LÓPEZ HERREROS, “Estudio de emisión de vıdeo sobre HTML5,” 2014.

[147] V. Wang, F. Salim, and P. Moskovits, *The definitive guide to HTML5 WebSocket*, vol. 1. Springer, 2013.

[148] Q. Liu and X. Sun, “Research of Web Real-Time Communication Based on Web Socket,” 2012.

[149] W. W. W. Consortium and others, “W3C. 2014. HTML 5: Techniques for providing useful text alternatives.” .

[150] ISO/IEC, “ISO/IEC 23009-1:2014 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats.” 2014.

[151] A. Hamza and M. Hefeeda, “A DASH-based free viewpoint video streaming system,” in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, 2014, p. 55.

[152] P. Lazarakis, “Dynamic adaptive streaming over HTTP,” 2014.

[153] S. Lederer, C. Mueller, C. Timmerer, C. Concolato, J. Le Feuvre, and K. Fliegel, “Distributed DASH dataset,” in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013, pp. 131–135.

[154] C. Alberti, D. Renzi, C. Timmerer, C. Mueller, S. Lederer, S. Battista, and M. Mattavelli, “Automated QoE evaluation of dynamic adaptive streaming over HTTP,” in *Quality of Multimedia Experience (QoMEX), 2013 Fifth International Workshop on*, 2013, pp. 58–63.

[155] I. Sodagar, “The mpeg-dash standard for multimedia streaming over the internet,” *IEEE MultiMedia*, no. 4, pp. 62–67, 2011.

[156] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, “Dynamic adaptive HTTP streaming of live content,” in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, 2011, pp. 1–8.

[157] J. K. Nurminen, A. J. Meyn, E. Jalonen, Y. Raivio, and R. Garcıa Marrero, “P2P media streaming with HTML5 and WebRTC,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, 2013, pp. 63–64.

[158] C. Alexandru, “Impact of WebRTC (P2P in the Browser),” *Internet Economics VIII*, vol. 39, 2014.

[159] J. Uberti and C. Jennings, “Javascript Session Establishment Protocol,” *draft-ietf-rtcweb-jsep-06 (work in progress)*, 2014.

[160] J. Rosenberg, “Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal for offer/answer protocols,” 2010.

[161] P. Rodrıguez Pérez, J. Cerviño Arriba, I. Trajkovska, and J. Salvachúa Rodrıguez, “Advanced Videoconferencing based on WebRTC,” 2012.

[162] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, “Session traversal utilities for NAT (STUN),” 2008.

[163] R. Mahy, P. Matthews, and J. Rosenberg, “Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun),” 2010.

[164] H. Alvestrand, “A Registry for WebRTC statistics identifiers,” 2012.