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 "***Diseño de una plataforma de agentes para control de servicios de video streaming móvil***" presentada por la doctoranda Doña Tatiana Gualotuña Álvarez y dirigida por la Doctora Doña Elsa María Macías López y el Doctor Don Álvaro Suárez.

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

Diseño de una plataforma de agentes para control de servicios de video streaming móvil

Autora

Tatiana Gualotuña Álvarez

Director(es)

Dra. Elsa Mª Macías López

Dr. Álvaro Suárez Sarmiento

El (Los) director(es)

El doctorando

Las Palmas de Gran Canaria, 30 octubre de 2015

Agradecimientos

Agradezco a mi padre celestial por su fortaleza en este camino lleno de aprendizajes, por su infinito amor y bendiciones. Un gracias desde lo profundo de mi alma a mi precioso hijo Sebastián, mi regalo divino, quien ha sido mi inspiración y alegría, quien ha sabido soportar mis ausencias y me ha enseñado con su gran ternura a ser valiente y luchar con todo el corazón para lograr la victoria.

También, agradezco a mi familia por su gran amor, por siempre estar a mi lado y compartir mis risas y llantos, por ser mi aliento en cada desánimo y la alegría en cada conquista. A mis amigos, por estar presentes en cada etapa de mi vida y por su motivación diaria para volver a soñar.

Con todo mi corazón, un sentido gracias a Elsa y Álvaro, por su paciencia y apoyo a levantarme de mis caídas y seguir perseverando en este hermoso anhelo.

No puedo olvidarte a ti Diego, amigo y hermano, compañero de esta travesía, donde mutuamente arrimamos nuestros hombros hasta lograr llegar a la orilla.

Resumen

La evolución de las comunicaciones inalámbricas y el desarrollo de los teléfonos móviles inteligentes han influido en una creciente demanda de aplicaciones para móviles, en especial video streaming. Los canales inalámbricos se ven afectados por el comportamiento impredecible de la Red, que puede generar quiebres de comunicación, que afectan la transmisión del video streaming. Se define un modelo básico basado en proxies que permite controlar una interrupción del video y posibilita al cliente la continuidad del video desde la posición en la que se encontraba una vez se restablezca el canal de comunicación.

Para su funcionamiento en entornos Web, se propone el incremento de funcionalidad a este modelo básico a través de agentes ontológicos que evalúan el estado de los recursos del dispositivo móvil y el uso de patrones software de diseño para establecer una posible interrupción y solicitar la ejecución de estrategias que permitan al cliente continuar visualizando el video desde el punto de quiebre. Este nuevo modelo se caracteriza por ser interoperable y multiplataforma.

En base a la experiencia obtenida, se planteó un interacción al modelo Web interoperable desarrollado, con miras a evaluar el estado de sensores para domótica controlados y permitir el despliegue del video streaming en tiempo real en el dispositivo móvil, esto se ha logrado manteniendo el modelo básico propuesto y reemplazando la orientación de los patrones software de diseño anteriormente seleccionados, cumpliéndose el objetivo del control de interrupción y continuidad del servicio de video streaming.

Se han logrado desarrollar las propuestas en base a un diseño iterativo del modelo básico el cual se mantiene constante, pero en sus incrementos se añaden patrones software y comportamientos, esto no supone un costo adicional en el tiempo de ejecución, más bien se ha logrado reducir el impacto negativo de una interrupción y mejorar la experiencia de usuario en los servicios de video streaming.

Palabras clave

Video streaming, comunicación inalámbrica, dispositivos móviles, agentes inteligentes, ontología, servicios Web, interrupción, recuperación, video streaming en tiempo real, sensores, ambientes domóticos.

Abstract

The evolution of wireless communications and the development of smart phones have influenced a growing demand of mobile applications, especially video streaming. Wireless channels are being affected by the unpredictable behavior of network, which can generate communication breakdowns that affect the video streaming transmission. A model based on proxies is defined so it controls a video interruption and allows the costumer the continuity video from the position where he was once that the communication channel is restored.

For operation in web environments, is proposed the functionality intends to this basic model through ontological agents that evaluate the status of mobile resources and the use of software design patterns to establish a possible interruption and request strategies execution that allow the costumer to continue viewing the video from the breaking point. This new model is characterized for being interoperable and multi-platform.

Based on the experience gained, an interaction with the web interoperable model developed was raised, in order to evaluate the sensors state for controlled home automation and allow the video streaming deployment in the mobile device in real time, this was achieved maintaining the basic model proposed and replacing the orientation of software design patterns previously selected, accomplishing the goal of interruption control and continuity service of video streaming.

It has developed proposals based on an interactive design of the basic model which is constant, but in it increases software patterns and behaviors have been added, this is not an additional cost in the execution time, rather it has managed to reduce the negative impact of an interruption and improve the user experience in video streaming service.

Keywords

Video streaming, wireless communication, mobile devices, smart agents, ontology, Web services, interruption, recovery, video streaming in real time, sensors, home automation environments.

Índices

Índice de contenido

Índice de tablas

Índice de Figuras

1. Introducción

En este capítulo se presenta la motivación y el problema que han influido en la realización de este trabajo de investigación y los objetivos de la presente tesis doctoral. Finalmente, se exhibe las aportaciones y la estructura de la memoria.

1.1. Motivación

La industria de medios de comunicación se ha convertido en un sector económico de alta importancia, donde video streaming es una de los servicios más populares ya que permite la transmisión de archivos multimedia a través del Internet. El uso del video streaming móvil es creciente en dispositivos inalámbricos ya que se caracteriza por demandar menores requisitos de hardware, solventando las limitaciones de los dispositivos inalámbricos como la capacidad de procesamiento y memoria.

El comportamiento impredecible de los canales inalámbricos genera interrupciones de comunicación, que afectan la transmisión del video streaming, esto puede provocar en el usuario el inicio de una nueva sesión o su abandono. Es necesario entonces, establecer mecanismos que identifiquen el restablecimiento de la comunicación y que permitan al cliente la visualización del video desde la posición en la que se encontraba, cuando ocurrió la interrupción.

Este problema ha sido estudiado en otros trabajos previos del *Grupo de Arquitectura y Concurrencia* (*GAC*) de la *Universidad de Las Palmas de Gran Canaria* (*ULPGC*). En esos trabajos se empleó el patrón de diseño software denominado *Proxy*. La principal motivación de este trabajo de tesis doctoral es analizar el modelo básico de video streaming sin proxies a través de patrones de diseño y extender mediante diseño basado en patrones diversas soluciones para mitigar en la medida de lo posible las interrupciones del servicio de video streaming móvil. Esto es, implantar este novedoso mecanismo en dispositivos inalámbricos de última generación como son los dispositivos *iPhone* [1] dicho mecanismo fundamenta su desarrollo en *Java Agent Development Framework* (*JADE*) [2] que es un framework que permite el desarrollo de agentes inteligentes para computadores de escritorio, teniendo una versión con menores exigencias, útil para dispositivos inalámbricos como es JADE *Lightweight Extensible Agent Platform* (*LEAP*) [3].

Buscamos experimentar y validar nuestras soluciones con la finalidad de lograr en el cliente la *Calidad de Experiencia* (*QoE*) [4], evaluando el nivel de calidad percibido y la valoración de la experiencia del servicio de video streaming en dispositivos inalámbricos.

1.2. Definición del problema y su contexto

La evolución de las tecnologías móviles y el desarrollo de las comunicaciones proporcionan una amplia disponibilidad de redes inalámbricas; sustentando este avance se ha optimizado el ancho de banda a través del incremento de la velocidad de las comunicaciones y la generación de códigos de compresión de datos.

En la actualidad los teléfonos móviles inteligentes presentan avances en memoria, capacidad de procesamiento, mayor calidad en la pantalla, disponen de interfaces de comunicación inalámbrica, todo este desarrollo ha influido en una creciente demanda de aplicaciones para móviles, uno de los cuales es video streaming.

La comunicación multimedia a través de la red inalámbrica incluye: voz, imagen y video; los canales se ven afectados por el comportamiento impredecible de la red, dado su elevada latencia, alta tasa de pérdida de datos y ancho de banda inestable [5]. Este comportamiento de los canales inalámbricos puede generar quiebres de comunicación, que afectan la transmisión del video streaming.

Las posibles interrupciones del canal influyen en el nivel de QoE que percibe el usuario, ya que puede causar la obligatoriedad de volver a visualizar el video desde el inicio o la decisión de abandonar la sesión por la percepción de varias interrupciones y el tiempo adicional de espera.

Basado en lo expuesto es importante proveer al cliente el uso de los servicios de video streaming que mantengan la continuidad y que gestionen efectivamente interrupciones de servicio; bajo esta premisa se busca definir mecanismos orientados a proporcionar control de desconexiones en ambientes inalámbricos específicamente en redes con tecnología *Wireless Fidelity* (*WiFi*) [6] enfocados principalmente en el despliegue del servicio del video streaming en dispositivos inalámbricos de última tecnología.

Otra premisa a lograr es generar un mecanismo tal que influya de manera directa en la percepción de QoE, al reducir el impacto negativo que causa una interrupción y aminorar los tiempos de ejecución de un video pese a las interrupciones que se hayan dado en el canal de comunicación.

1.3. Objetivos

El objetivo principal de este trabajo es demostrar que el mecanismo de control de interrupciones basado en proxies, el cual se fundamenta en patrones software de diseño, provee continuidad del servicio de video streaming en los dispositivos inalámbricos al ocurrir interrupciones, influyendo de manera positiva en la QoE del usuario al eliminar las retransmisiones que ocasionan mayores tiempos de ejecución del video y congestión del canal inalámbrico.

El segundo objetivo es generar una solución básica que utilice el mecanismo de control de interrupciones para iOS, obteniéndose un aplicativo novedoso denominado JADE-iPhone que controla interrupciones sin exigir al usuario el reinicio de una sesión RTSP y pueda recibir en diferido los flujos de vídeo que son temporalmente almacenados en un buffer hasta que el dispositivo móvil vuelva al área de cobertura.

Se establece como tercer objetivo el diseño de un sistema web interoperable de video streaming que gestione las interrupciones mediante agentes que son transformados a servicios Web, encargados de evaluar el estado de los recursos del dispositivo móvil y predecir una eminente interrupción inalámbrica en base a las reglas ontológicas establecidas y reducir el impacto negativo que ocasiona una interrupción en el usuario al proporcionar el servicio de reconexión automático del video streaming desde la última posición almacenada, permitiendo la continuidad de la reproducción.

Un último objetivo es reemplazar las funcionalidades adicionadas a la solución básica con miras a que el mecanismo de control de interrupciones pueda actuar en la transmisión de video streaming en tiempo real en ambientes domóticos controlados, basado en sensores que son evaluados de manera ontológica y reducir el impacto negativo que ocasiona una interrupción de video streaming en el usuario, ya que proporciona el servicio de reconexión automático y la grabación del video streaming bajo demanda de lo que sucedió durante la interrupción, para que pueda ser visualizado cuando el cliente lo requiera.

1.4. Contribuciones

Para llevar a cabo los objetivos planteados se han realizado las siguientes contribuciones:

- Se ha establecido una solución básica que fundamenta su diseño en patrones software, logrando caracterizarse por ser modular, flexible, escalable… cumpliendo de forma efectiva el control de interrupciones al aplicar observadores encargados de evaluar el almacenamiento y el canal de comunicación para determinar una interrupción y actuar de forma proactiva mediante el almacenamiento temporal de los frames y la reproducción de estos en el cliente una vez se haya dado la reconexión. Los resultados experimentales a través del desarrollo de un aplicativo JADE-iPhone, muestran el efecto positivo de esta solución al eliminar los tiempos de retransmisión y de reinicio de sesión, esto influye en la descongestión del canal de transmisión y en mejores tiempos de ejecución del servicio de video streaming y por ende provee mejor QoE al usuario. Los productos *iPhone Operating System* (*iOS*) [7] se crean bajo una arquitectura propietaria, los aplicativos que pueden implantarse en este tipo de móviles se desarrollan en el lenguaje Objective C [8]. Nuestro interés es determinar la arquitectura *Modelo Vista Controlador* (*MVC*) [9] que refleje el funcionamiento del modelo básico, analizar a profundidad los componentes que lo conforman y sus interrelaciones, para luego realizar una serie de experimentaciones tendientes a definir su validez y los beneficios que se obtienen al evitar visualizar un video desde su inicio si se diera una interrupción.
- Con la finalidad de proveer interoperabilidad y proactividad en el control de una interrupción al transmitir video streaming, se desarrolló la solución Web interoperable, que en base a agentes con comportamientos ontológicos determina una posible interrupción al analizar el estado de los recursos del dispositivo móvil: batería, latencia y geolocalización y traduce las acciones de los agentes en servicios Web para publicarlos en la red. Con la experimentación práctica se ha demostrado que es factible mantener el modelo básico e incrementar nuevas funcionalidades, el tiempo que se utiliza

para la ejecución de estas nuevas funcionalidades se solapan con el tiempo de visualización del video, también es necesario indicar que al ocurrir interrupciones en la transmisión del video streaming, el control proactivo que realiza esta solución apoya en la reducción del tiempo de ejecución del video, de forma similar que lo consiguió el aplicativo JADE-iPhone. Nuestro interés es aplicar la metodología incremental e iterativa, por lo que se mantuvo el modelo básico MVC y se adicionaron nuevas funcionalidades basados en el uso de otros patrones software. Para la implementación experimental del modelo propuesto, se ha utilizado un add-on de JADE que es *Web Services Integration Gateway* (*WSIG*) [10] para lograr la transformación de agentes a servicios Web y ser consumidos en el entorno Web, el agente creado realiza el análisis ontológico del estado de los recursos del dispositivo inalámbrico como son batería, latencia y geolocalización para establecer la posibilidad de una interrupción, permitiendo almacenar proactivamente la posición del video, información muy útil una vez que se supere el problema de comunicación.

- Para evaluar la efectividad del mecanismo de control de desconexiones en servicios de video streaming en tiempo real, se estableció una solución que analiza del estado de los sensores en el ambiente controlado (movimiento, temperatura y luminosidad) mediante agentes interoperables que definen alerta de seguridad y activan la cámara cuya captura es desplegada en el dispositivo móvil en tiempo real, esta propuesta mantiene la continuidad del video en caso de que existan interrupciones, mediante un bucle de reconexión encargado de restablecer automáticamente el servicio. Mientras dure la interrupción el video es alojado en un almacenamiento temporal, para posibilitar su visualización cuando el cliente lo requiera. Con la experimentación práctica se ha demostrado que es factible mantener el modelo básico e incrementar o reemplazar nuevas funcionalidades, sin afectar el tiempo de ejecución del video streaming que soporte interrupciones, proveyendo de buenos niveles de QoE al proporcionar al usuario continuidad del servicio en tiempo real con reconexiones automáticas en caso de interrupciones y posibilidad de revisar en diferido los

videos almacenados de lo que ocurrió durante la interrupción dada. Los sensores constantemente monitorean el ambiente controlado, en caso de intrusos definen una alerta que activara la cámara generando video streaming en tiempo real a ser transmitido al dispositivo móvil. Estamos interesados en implantar el mecanismo de control de interrupciones en el video streaming en tiempo real con el objetivo de consolidar la validez del modelo básico propuesto y definir que la arquitectura establecida adicional puede ser implementada con cualquier otro recurso observable.

En cuanto a publicaciones internacionales, 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*,* donde se expone una arquitectura basado en proxies y agentes inteligentes para dispositivos inalámbricos Android, que se encarga de resolver la interrupción en ambientes inalámbricos y reanudar automáticamente la sesión de video streaming sin pérdida de la información al aplicar la técnica de almacenamiento intermedio.

En cuanto a ponencias internacionales, este trabajo ha dado lugar a la ponencia:

- *G. Raura, T. Gualotuña, J. Báez, M. Ñauñay*, ***Renderización de imágenes panorámicas de alta resolución en dispositivos móviles mediante el uso de agentes de software***, *5to Congreso Internacional de Tecnologias, Contenidos Multimedia y Realidad Virtual*, 2011, La Habana, Cuba. Propone un mecanismo de control de interrupciones en enlaces inalámbricos no seguros, que permita la renderización eficiente de fotografías panorámicas de alta resolución al recuperar la sesión automáticamente.

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

- ***La Web semántica orientada a la generación de aplicaciones para discapacitados***, que promueve el desarrollo de aplicaciones especializadas en contenidos para discapacitados utilizando los principios de la Web semántica a través del uso de ontologías, agentes de software y herramientas.
- ***Diseño de una plataforma de agentes para control de interrupciones para servicios de video streaming en dispositivos móviles***, que permitió el desarrollo de software en Android para recuperar la sesión de streaming frente a una interrupción y garantice la recepción completa del video.

En relación a la definición y propuestas de proyectos de investigación a ejecutarse se ha dado lugar a:

- ***Extracción de conocimiento de un invernadero mediante el uso de Wireless Sensor Network*** (***WSN***) ***y tratamiento de flujos de datos utilizando técnicas adaptables a cambios del entorno***, que busca desarrolla un sistema que permita capturar la información de la humedad relativa, temperatura y suelos por medio de sensores en un invernadero, los cuales serían procesados y analizados por medio de la minería de datos con la finalidad de establecer patrones que apoyen en la toma de decisiones.

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***, que busca mejorar la comprensión del proceso de desarrollo de software en la industria para lo que se realizaron experimentos controlados y estudios empíricos que permiten brindar conocimientos sobre el comportamiento de las tecnologías de software en diferentes entornos.

1.5. Estructura de la memoria

En la prestación del servicio de video streaming móvil en ambientes inalámbricos es necesario proporcionar al usuario de una buena experiencia, para ello se ha desarrollado varias propuestas que buscan mitigar el impacto de una interrupción. El trabajo que ha sido realizado se presenta en los seis capítulos de esta memoria.

En el capítulo 2 se establece los avances de las redes inalámbricas así como la evolución de los dispositivos inalámbricos, lo que ha influido en el apogeo de servicios multimedia como es video streaming. Pese a este desarrollo se definen problemas que presenta la creciente demanda de los servicios y los problemas que enfrentan la comunicación al tener que solventar mayor capacidad y velocidad en la transmisión, esto confluye en la generación de soluciones que permitan aminorar los impactos de las desconexiones en el cliente al utilizar servicio de video streaming.

En el capítulo 3 se expone a detalle la implementación del mecanismo que controla interrupciones, de forma puntual, se explica la generación del software JADE-iPhone que implementa la solución básica propuesta, el cual se basa en agentes inteligentes para controlar una interrupción cuando se transmite un video al dispositivo móvil iOS y permite la continuidad del video una vez que se reanuda la comunicación inalámbrica.

En el capítulo 4 se establece una solución web interoperable, que busca lograr que el servicio de video streaming proporcionado a cualquier dispositivo móvil, pueda enfrentar el comportamiento caótico de la transmisión inalámbrica al controlar una interrupción mediante su predicción ontológica, proporcionar la reconexión automática y el despliegue del video desde el punto en el que se encontraba al ocurrir la interrupción.

En el capítulo 5 se plantea una solución que apoya en la captura y análisis de la información proporcionada por sensores de luz, temperatura y movimiento, para establecer de forma ontológica una alerta en ambientes domóticos controlados, al suscitarse eventos inusuales el cliente mediante dispositivos inalámbricos puede observar el video en tiempo real de lo que está sucediendo en el lugar controlado, en

caso de ocurrir una interrupción se ejecuta un bucle de reconexión que solicita el enlace automático con la cámara una vez resuelto el problema de la interrupción.

En el capítulo 6 se presentan las conclusiones y líneas de trabajo futuro.

2. Problemática de video streaming en dispositivos inalámbricos

En este capítulo se presenta el desarrollo de las redes inalámbricas en especial redes WiFi; la evolución de los dispositivos inalámbricos con énfasis los de última generación; la creciente demanda de servicios multimedia y el apogeo de los servicios de video streaming. Se establece también la problemática que enfrenta esta evolución tecnológica como es la saturación de los canales inalámbricos y su comportamiento impredecible. Se definen las dificultades que ocasionan las interrupciones en la prestación del servicio de video streaming y los retos propuestos con miras a plantear mecanismos que permitan reducir el impacto de una interrupción en el cliente.

2.1. Tecnologías de redes y dispositivos inalámbricos

La tecnología inalámbrica es un sistema de comunicación que ha experimentado creciente popularidad al no exigir cableado en su implementación y posibilitar al usuario movilidad en tiempo real lo que apoya a su productividad y proporciona oportunidades de servicio imposibles de lograr con las redes cableadas. Esta nueva infraestructura de comunicación presenta ventajas como: movilidad, flexibilidad, escalabilidad, velocidad, simplicidad y costos reducidos de instalación.

El avance de las redes inalámbricas ha permitido el desarrollo de muchos servicios y aplicaciones como: mensajes multimedia, *Voz sobre Internet Protocol* (*VoIP*), telefonía IP... La convergencia de las aplicaciones inalámbricas con los sistemas móviles influyen en la evolución del ancho de banda móvil inalámbrico, son cada vez mayores la demanda de aplicaciones multimedia a ser ejecutadas en los teléfonos móviles inteligentes.

En [11] se determina que el volumen de tráfico generado en redes con tecnología WiFi alcanzó un 49% del tráfico total de Internet en el 2012, mientras que en el 2017 se prevé su incremento al 56%. En relación a las redes móviles, en [12] se establece que el tráfico de este tipo de redes supondría un 12% del tráfico total de Internet para el 2017, avizorando su crecimiento 3 veces más rápido que el tráfico de las redes fijas. Con lo expuesto, parece evidente que los próximos años se dará un crecimiento de las redes inalámbricas ya sean tecnologías WiFi o redes móviles y un decrecimiento del uso de redes cableadas. En resumen para el año 2017 el tráfico de las redes móviles en combinación con las redes WiFi alcanzará un total del 68% del tráfico total de la red.

En la Figura 2.1 (adaptada de [13]) puede revisarse la estimación para el 2019, el 54% del tráfico generado por dispositivos inalámbricos se realizará por medio de conexiones WiFi y un 46% lo haría a través de las redes móviles.

Se presenta la Figura 2.2 (adaptada de [14]) en la que se observa el amplio crecimiento que experimentarían las tecnologías inalámbricas para el 2020, considerando el volumen de tráfico en Internet y el número de usuarios que utilizarían dispositivos inalámbricos.

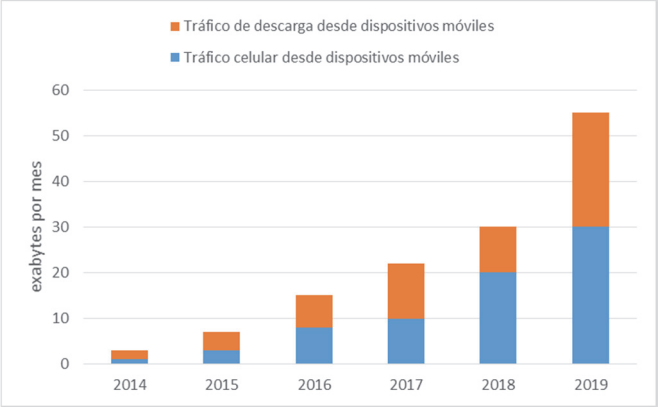

Figura 2.1 Evolución del tráfico de dispositivos inalámbricos

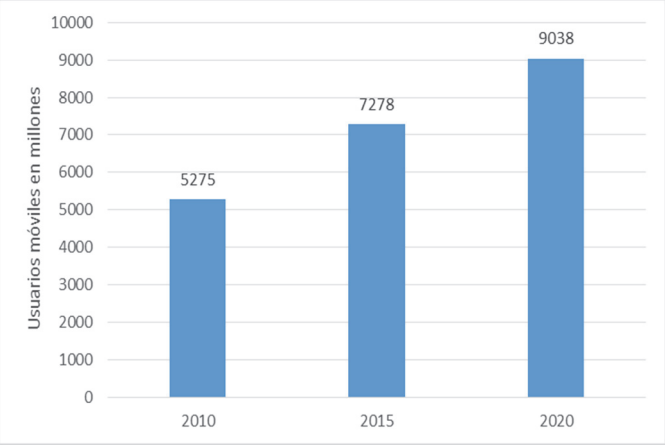

Figura 2.2 Número de usuarios móviles en el mundo

El crecimiento explosivo de los teléfonos inteligentes (*smart*) y otros dispositivos inalámbricos ha creado un tsunami inalámbrico ya que la característica más importante de estos equipos es el acceso a Internet. Aunque hay múltiples tecnologías disponibles para el acceso a Internet, la tecnología WiFi, es la función utilizada por defecto en la mayoría de los dispositivos inalámbricos. Actualmente, se presenta un alto incremento de dispositivos clientes que demandan anchos de banda para ejecutar aplicaciones como video streaming móvil. Es constante la creciente necesidad de optimización del rendimiento y uso del sistema WiFi; para satisfacer estas demandas este estándar ha estado en constante evolución para lograr un mayor rendimiento de datos y eficiencia espectral.

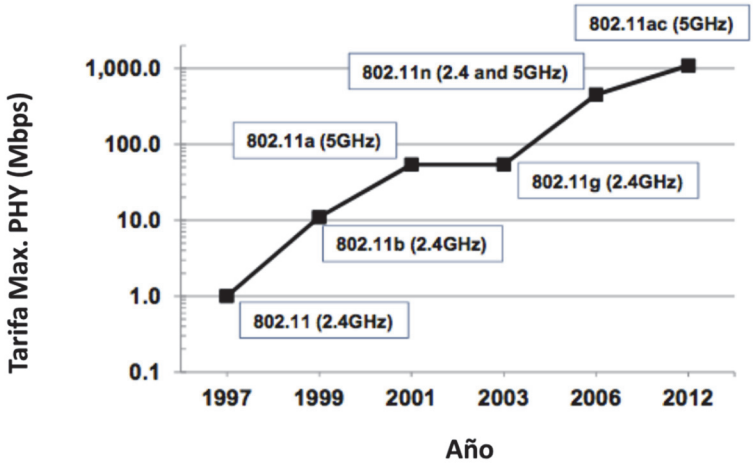

Figura 2.3 Evolución del estándar IEEE 802.11

Tabla 2.1 Análisis de los estándares de la norma 802.11

	No. of flujos	Ch.BW (MHz)	Modulación and Codificación	Req EVM	Tasa de Datos Max. PHY
802.11 a	1	20	64-QAM, 3/4	-25	54 Mbps
802.11n	2	20	64-QAM, 5/6	-27	144 Mbps
802.11n	2	40	64-QAM, 5/6	-27	300 Mbps
802.11ac	3	80	256-QAM, 5/6	-32	1.3 Gbps
802.11ac	2	160	256-QAM, 5/6	-32	1.73 Gbps

La evolución del estándar *Instituto de Ingeniería Eléctrica y Electrónica* (*IEEE*) 802.11 de la red local inalámbrica WiFi, puede observarse en la Figura 2.3 y tabla 2.1 (adaptadas de [14]), con particular interés el desarrollo del estándar IEEE 802.11n, el cual aparece por el año 2006 y presenta la posibilidad de operar a 2.4 GHz y 5 GHz.

Las nuevas redes inalámbricas de 5 GHz operan en un espectro más amplio con mayor número de canales no compartidos y pueden ser combinados a mayor velocidad, buscan evitar los continuos problemas de saturación de las redes [15].

Las ventajas de las redes WiFi son las siguientes:

- Permiten a los usuarios beneficiarse de la movilidad, al poder comunicarse o acceder a la información en tiempo real y utilizar los servicios proporcionados por la red, sin importar el lugar donde se encuentre.

- La tecnología inalámbrica ofrece facilidad de instalación al reducir los esfuerzos de pasar los cables a través de techos y paredes, y proporcionan menores costos de instalación.
- Posibilitan el acceso a los recursos desde cualquier ubicación, tiempo y lugar.
- Las configuraciones inalámbricas permiten una fácil incorporación de cambios como creación de nuevos usuarios, ampliaciones de red.

También, las redes WiFi presentan desventajas como:

- Las redes WiFi presentan menor ancho de banda en relación con las redes cableadas.
- Este tipo de redes son más susceptibles a la interferencia de factores externos climáticos y electromagnéticos, también pueden ocurrir interferencias con otros dispositivos que estén trabajando en la misma banda de frecuencia; lo que genera mayores tasas de error en la comunicación.
- Presentan menor potencia, distancia y velocidad que las redes cableadas. Hay varios organismos que están trabajando en estándares que mejoren este factor.
- El medio inalámbrico ofrece menor confiabilidad que el medio físico cableado, ya que la señal está más expuesta a ser captada por terceros.

Las futuras redes inalámbricas soportaran Gbps para la comunicación multimedia entre las personas y dispositivos con una alta fiabilidad y una cobertura uniforme. La escasez del espectro se aliviara por los avances en radios cognitivas, algoritmos de eficiencia de energía y hardware que ayudan a generar sistemas inalámbricos ecológicos [16] [17]. Podrían hacer converger tecnologías heterogéneas [18] y cambiar hábitos sociales [19].

La evolución del teléfono móvil ha permitido disminuir su tamaño y peso, tienen mayores prestaciones: desarrollo de baterías más pequeñas y de mayor duración, pantallas más nítidas y de colores, mayor capacidad de memoria y bufferización. Los servicios móviles de banda ancha a nivel mundial se han incrementado gracias a las características tecnológicas que presentan los dispositivos inalámbricos para la comunicación [20] [21] haciendo posible la *Internet móvil* [22].

Tabla 2.2 Evolución de los dispositivos inalámbricos iPhone

MODELO	CARACTERÍSTICAS
iPhone original (**2007**)	Con sistema operativo i*OS* 2.0, pantalla táctil de 3,5 pulgadas, resolución de 480x320, acelerómetro, Bluetooth, almacenamiento de 4 Gb y 8 Gb, WiFi y un botón principal, revolucionó la industria por su diseño y prestaciones del software como: integración con Google Maps, iTunes con coverflow, Safari, el estado del tiempo y la posibilidad de subir video desde el teléfono a YouTube. Sin embargo, presentó algunas debilidades como, conexión *Enhaced Data Rates* (*EDGE*) en lugar de 3G, cámara de 2 Mpx contra 5 Mpx, características que ya exhibían los demás celulares de gama alta.
iPhone 3G (**2008**)	Agrega compatibilidad con las redes 3G, introdujo mejoras en el conector de auriculares, *Global Positioning System* (*GPS*), batería y más aplicaciones. Su precio disminuyó lo que aumentó su popularidad.
iPhone 3GS (**2009**)	Actualizó su procesador, gráficos, conectividad, Bluetooth, añadió cámara de 3 Mpx con grabación de video, brújula, grabación de voz, e integró el software OS 3.0 con teclado horizontal y la función de "cortar y pegar", con capacidad de memoria 16Gb y 32 Gb.
iPhone 4 (**2010**)	Pantalla con acabado de cristal brillante, procesador A4, cámara frontal para llamadas FaceTime, cámara principal de 5 Mpx con flash *Light Emitting Diode* (*LED*), grabación HD a 720p, giroscopio y mejor batería, además de una barra lateral de acero con las antenas incluidas.
iPhone 4S (**2011**)	Procesador A5 de doble núcleo y dos antenas para mejorar la recepción de la señal, la calidad del sonido y la velocidad de descarga de datos, iOS, iCloud y cámara de ocho megapíxeles con un lente que permite grabar videos, grabación Full HD a 1080p. Su batería tiene una duración de hasta ocho horas de conversación, nueve horas de navegación WiFi y hasta 40 horas si se escucha música. Lo más destacable es su sistema de reconocimiento de voz llamado Siri.
iPhone 5 (**2012**)	Dispositivo con pantalla de 4 pulgadas, 20% más liviano, incluye chip A6, soporte para conectividad 4G *Long Term Evolution* (*LTE*), tres micrófonos, funciones de panorama, nuevo conector 80% más pequeño, reversible y más veloz, con sistema iOS 6.
iPhone 5S (**2013**)	Nueva función de reconocimiento dactilar, diseñado para aumentar la seguridad del aparato. Posee procesador A7, cámara con grabación lenta, con sistema iOS 7.
iPhone 6 (**2015**)	Mide 5.5 pulgadas, presenta pantalla de 4.7 pulgadas, display HD, *chips Near Field Communication* (*NFC*) y Apple Play.

Desde el primer teléfono portable de los años 1940, hasta el primer teléfono móvil de los años 1990 hasta la actualidad se ha producido un avance espectacular en tecnología y comunicación. El nacimiento del primer iPhone supuso una innovación disruptiva de consecuencias importantes. En la tabla 2.2 se detalla la evolución de estos dispositivos.

2.1.1. Sensores inalámbricos domóticos

Un sensor es un dispositivo para detectar y señalar una condición de cambio. Con frecuencia una condición de cambio se trata de la presencia o ausencia de un objeto o

material (detección discreta). También puede ser capaz de medirse, como un cambio de distancia, tamaño o color (detección analógica). Los sensores posibilitan la comunicación entre el mundo físico y los sistemas de medición y/o control, tanto eléctricos como electrónicos, utilizándose extensivamente en todos los procesos industriales y no industriales para propósitos de monitoreo, medición, control y procesamiento. Un sensor se caracteriza por ser capaz de realizar procesamientos gracias al microprocesador, almacenar información en su memoria y la capacidad de captar energía que alimenta al sensor. En base al desarrollo de las comunicaciones inalámbricas surgen nuevos servicios que pueden llevarse a cabo sin la interacción directa con el ser humano a través de sensores. Existen diferentes tipos de sensores: temperatura, luminosidad, de presión, de humedad, de velocidad, de aceleración, de presencia, de volumen… al adicionar a estos sensores capacidad de comunicación inalámbrica y conformación de redes se posibilita la obtención de información valiosa a través de redes de sensores inalámbricos, su demanda es creciente en aplicaciones de seguimiento, seguridad, salud, gestión de la energía, automatización de tareas domésticas, ocio y entretenimiento, operación y mantenimiento de instalación, formación y cultura…

Los sensores son capaces de integrarse con las redes de comunicación de una manera rápida y transparente [23] [24]. El desarrollo de las redes posibilita la creación de ambientes, donde los sensores juegan un rol importante para la percepción, captación y distribución de la información obtenida a partir de un evento o fenómeno ambiental [25] [26].

Los sensores son autónomos, añaden valor a los datos para dar soporte a la toma de decisiones y al ser conectados, constituyen un sistema distribuido que trabaja de forma cooperativa para medir variables físicas y cambios en condiciones ambientales [27]. El papel de los sensores es apoyar en:

- Identificación y seguimiento a los actores del entorno.
- Definición del intervalo de tiempos para las acciones de localización y el contexto del objeto.
- Reconocimiento de actividades e interacciones.

- Asociación de las acciones con la semántica de la acción identificando tareas o patrones software de comportamiento.

La tecnología de sensores se puede categorizar en intrusiva y no intrusiva. Los sensores intrusivos son aquellos que requieren ser instalados directamente en la superficie del lugar, por lo que es necesario excavar la misma, los sensores no intrusivos requieren estar fijados en el techo, suelo o pared, presentando una instalación sencilla.

Los sensores se pueden categorizar de acuerdo al tipo de señal de salida. a) Los analógicos dan como salida un valor de tensión o corriente, en forma continua dentro del campo de medida. b) Los digitales dan como salida una señal en forma de una palabra digital. c) Los sensores todo-nada indican cuando la variable detectada rebasa un cierto umbral.

En la tabla 2.3 se define los tipos de sensores. Las redes de sensores pueden ser aplicados para diversos servicios, en especial para la seguridad en ambientes domóticos, sin embargo la baja capacidad de procesamiento, memoria y energía de los sensores motivan a plantear comportamientos semánticos que basados en reglas ontológicas, analicen el entorno y definan acciones frente a problemas como detección de intrusos, alerta de incendio… [28] [29] [30].

Tabla 2.3 Tipos de sensores

TIPO	DESCRIPCIÓN
Sensores inductivos	Los sensores inductivos tienen una distancia máxima de accionamiento que depende en gran medida del área de la cabeza censora, se utilizan principalmente en aplicaciones industriales. Detectan cualquier objeto metálico sin necesidad de contacto: control de presencia o de ausencia, detección de paso, de atasco, de posicionamiento, de codificación y de contaje.
Sensores capacitivos	Son adecuados para detectar objetos o productos no metálicos de cualquier tipo (papel, vidrio, plástico, líquido).
Sensores fotoeléctricos	Detectan un objeto por medio de un haz luminoso. Sus dos componentes básicos son un emisor y un receptor de luz. Dependiendo del modelo de detector, la emisión se realiza en infrarrojo o en luz visible verde o roja. Emplean dos procedimientos para detectar objetos: • Bloqueo del haz: En ausencia del objeto el haz luminoso alcanza el receptor. Un objeto bloquea el haz al penetrar en él. El sensor de barrera emplea este procedimiento. • Retorno del haz: En ausencia del objeto, el haz no llega al receptor. Cuando un objeto penetra en el haz, lo envía al receptor. El sensor de proximidad emplea este procedimiento

Debido al impulso y crecimiento de la domótica, es necesario disponer de un medio de representación del conocimiento, la introducción de ontologías en las redes de sensores permite la construcción de ambientes semánticos, que analizan el entorno y toman decisiones [31].

Una red de sensores posee una infraestructura de comunicación que permite almacenar datos, notificar alguna condición específica y hacer medición y seguimiento de variables físicas en un entorno determinado [32].

Se puede medir: temperatura, humedad, presión, dirección y velocidad del viento, intensidad de iluminación, vibración, intensidad del sonido, voltajes en líneas de potencia, concentraciones químicas, niveles de contaminación, funciones vitales del cuerpo, concentraciones de gases… [33].

2.1.2. Dispositivos inalámbricos empotrados

Arduino es [34] una placa que lleva un microcontrolador Atmel AVR y ofrece al usuario varios puertos de entrada/salida y un entorno de desarrollo que apoya en los proyectos vinculados al mundo de la electrónica y los microcontroladores de forma sencilla que no plantea barreras económicas o tecnológicas. *Arduino* ya sea por separado o combinado con *Raspberry Pi* [35] es la base de sistemas que permiten automatizar procesos en los hogares y, por tanto, desarrollar sistemas domóticos de bajo coste. Controlar la iluminación de la casa desde una tableta o teléfono móvil, subir o bajar las persianas son algunas de las cosas que se pueden automatizar en los hogares.

Actualmente hay una gran cantidad de modelos de plataformas Arduino con diferencias en cuanto a características y posibilidades, números de entradas/salidas, microcontrolador… pero compatibles entre sí. Para facilitar su uso se desarrolló simultáneamente con la plataforma Arduino un *Integrated Development Environment* (*IDE*) en el que se usa un lenguaje de programación parecido a C++, basado en el lenguaje Wiring, el entorno de desarrollo está basado en Processing. El IDE permite editar, compilar y enviar el programa a la plataforma Arduino, comunicarse vía serialmente y

mostrar los datos en una ventana terminal. La plataforma Arduino se comunica con el IDE mediante un programa cargador, precargado en el microcontrolador.

Con Arduino se puede tomar información del entorno a través de sensores conectados a sus entradas analógicas y digitales, puede controlar luces, motores y otros actuadores directamente o a partir de las señales de control generadas en sus salidas, ofrece algunas ventajas como:

- Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras.
- El software Arduino se ejecuta en sistemas operativos Windows, Macintosh y Linux.
- El entorno de programación es simple y claro, fácil de usar.
- El software Arduino está publicado como herramientas de código abierto, el lenguaje puede ser expandido mediante librerías C++.
- El Arduino está basado en microcontroladores, los planos para los módulos están publicados bajo la licencia Creative Commons.

Raspberry Pi es un computador de bajo coste y su tamaño es de 8,5 x 5,3 cm, su sistema de arranque es una tarjeta de memoria *Secure Data* (*SD*) y su capacidad puede variar según las aplicaciones que se le quiera dar. La alimentación se realiza mediante una conexión micro USB. En la placa se encuentra salida de video y audio a través de un conector *High Definition Multimedia Interface* (*HDMI*), permitiendo la conexión a televisores, monitores. Incluye conectores de expansión *General Purpose Input/Output* (*GPIO*) que pueden ser utilizados para la comunicación con otros dispositivos, como módulos específicos de expansión, e incluso es compatible con Arduino.

Existen dos modelos de Raspberry Pi, el modelo A y B, que comparten una serie de características, como el chip, el procesador gráfico, las entradas/salidas y la capacidad para reproducir vídeo. El modelo B tiene el doble de memoria, 512 MB por 256 del A, tiene dos USB, por uno del modelo A y conector para cable de red. Dispone de un procesador a 700 Mhz. y puede acelerar gráficos 3D por hardware, es decir, se

podía comparar con los ordenadores que salieron al mercado en el 2003, con la gran diferencia de tamaño, peso y con un consumo 80 veces menor.

Raspberry Pi es 40 veces más rápido que un Arduino cuando se trata de velocidad de reloj. Además, Pi tiene 128,000 veces más memoria RAM.

2.2. Video streaming inalámbrico

Los avances en comunicación inalámbrica proporcionan al usuario el acceso a información en tiempo real como noticias, alertas médicas, mensajes de advertencia de desastres naturales, publicados por usuarios u organizaciones en cualquier parte del mundo; pueden interactuar con otras personas en el ciberespacio a través de redes sociales; buscar información publicitaria de interés para determinar la ubicación de: restaurantes, hoteles... [36] [37] [38]. Con el rápido desarrollo de las tecnologías inalámbricas y la popularidad de los clientes móviles inteligentes, video streaming se ha convertido en una de las más populares aplicaciones durante los últimos años.

Streaming es una técnica que consiste en la fragmentación y compresión del video, se caracteriza por ser un flujo continuo de datos, los cuales se reproducen y descargan simultáneamente, sin la necesidad de almacenar toda la información en la memoria del cliente. La velocidad a la que se consumen los datos se ajusta a la velocidad de transmisión; este acoplamiento se logra gracias al buffer receptor, el cual permite la compensación de posibles variaciones que pueden darse en la transmisión, si el buffer se vacía el servicio se interrumpe.

La técnica de video streaming permite que los objetos multimedia se reproduzcan y descarguen simultáneamente desde el Internet sin la necesidad de almacenar toda la información en la memoria del cliente. La ventaja de esta técnica es que no requiere que el vídeo se almacene en el teléfono móvil dejando espacio para albergar memoria auxiliar y programas para tratar el vídeo de manera eficiente. Por otro lado, gracias a que no se consume esa cantidad de memoria, se pueden desplegar programas eficientes y poderosos en los terminales móviles.

El proceso de video streaming se inicia desde el cliente, conectándose con el servidor que comienza a mandar el archivo. Para el envío se utiliza un *Compresor-Descompresor* (*CODEC*) que comprime el archivo que viaja sobre un protocolo ligero. En el cliente se constituye un buffer que almacena la información que va llegando. Una vez que el buffer contiene una parte del archivo comienza la reproducción del mismo, al mismo tiempo que continúa con la descarga [39].

Cabe destacar la importancia de la sincronización en este tipo de sistemas, ya que los programas no reproducirán el archivo. Si hay variación de retraso en la conexión, se podría reproducir el contenido del buffer, según el protocolo. Si se sufren demasiados retrasos, el buffer se vacía y la ejecución del archivo también se cortaría [40] [41].

De acuerdo a la Figura 2.4, la transmisión de video streaming exige la presencia de dos canales diferenciados: por un lado el flujo de control utilizado para gestionar el establecimiento de la comunicación y por otro lado el flujo para envió de datos multimedia que se constituye el canal de transporte de los frames.

Para iniciar la transmisión de los contenidos multimedia a través de la red de datos, la información debe segmentarse en frames de tamaño adecuado. El cliente recibe los frames y procede a almacenarles en un buffer, cuando este se llena comienza la reproducción, mientras tanto el buffer continua recibiendo nuevos frames.

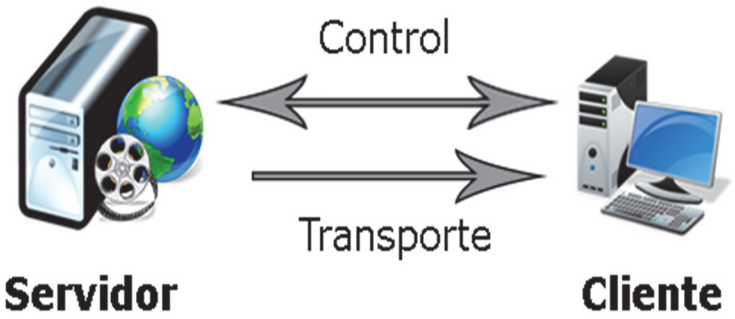

Figura 2.4 Servicio de video streaming

Este buffer del cliente se caracteriza por aminorar los efectos de variaciones en las condiciones de la red, al producirse un descenso en el ancho de banda, el cliente puede continuar con la reproducción de los paquetes alojados en el buffer, en caso de que los problemas de comunicación persistan, el buffer llegara a vaciarse y la reproducción se detendrá.

Los tipos de servicio de video streaming son:

- *Bajo demanda*: el video es almacenado en un servidor de video streaming, el cual está en capacidad de atender las peticiones de visualización del contenido, entregándolo mediante flujos para ser visualizado en el reproductor del cliente, el usuario tiene la posibilidad de controlar el flujo para detener su ejecución, realizar un retroceso, una pausa o pasar a otra escena.
- *En tiempo real*: servicio de transmisión en tiempo real, orientado a la multidifusión, sin interactividad. Los usuarios acceden a un único flujo de información en el instante temporal en el que se encuentre la emisión.

La arquitectura básica en la que se basan los servicios de video streaming es la arquitectura cliente servidor. De acuerdo a la Figura 2.5 se puede observar los componentes que son explicados a continuación:

- *Sistema de producción*: se encarga de capturar los contenidos en vivo a través de cámaras o micrófonos para transmitirlos en el formato adecuado al servidor. Este sistema está conformado por elementos tanto de hardware y software que tienen la funcionalidad de comprimir los contenidos multimedia y darles un formato adecuado, con la finalidad de que resulten aptos para su transmisión a través de los protocolos de transporte. En la transformación de los contenidos originales participan los CODEC de audio y/o video así como las tasas de codificación.
- *Sistema de almacenamiento*: se encarga de almacenar los contenidos, para que puedan ser solicitados por los clientes. Este componente es utilizado para video bajo demanda.

- *Servidor*: se encarga de recibir y procesar peticiones de los clientes; establecer la sesión mediante el intercambio de una serie de parámetros y procesar el envío de los contenidos multimedia.
- *Reproductor*: se ubica en el cliente y se encarga de la interacción con el servidor, realiza actividades como: establecer conexiones, procesar los frames y reconstruir el flujo de datos recibido.
- *Proxy*: es un proxy multimedia que se encarga de mantener la comunicación con el cliente durante la reproducción del video, mantiene el estado de la sesión e interactúa con el usuario.

En la telefonía móvil, video streaming está sometido al comportamiento de los canales, existiendo problemas externos como interferencias que provocan desconexiones en la transmisión que muchas veces son molestas para el usuario y hacen que la técnica de video streaming pierda efectividad.

Es por esto que para recibir los datos de video streaming se debe tener una aplicación dedicada llamado reproductor o media player, que debe ofrecer las siguientes características:

Figura 2.5 Arquitectura básica del servicio de video streaming

- El reproductor debe ser capaz de descomprimir los datos de audio y video a medida que vayan llegando.
- Debe ser capaz de eliminar el jitter antes de enviar los paquetes.
- Corrección de errores para poder evitar la pérdida de paquetes.

El tráfico de video streaming a través del Internet ha ido creciendo a un ritmo exponencial debido a la abundante disponibilidad de contenido de video en el Internet y la amplia disponibilidad de los dispositivos inalámbricos con altas capacidades de visualización en el usuario. Este desarrollo ha influido en el surgimiento de aplicaciones de video streaming como *Internet Protocol Television* (*IPTV*), video conferencia, educación en línea..., las cuales imponen requisitos de ancho de banda, demora y latencia [42] [43].

En un canal inalámbrico, el mayor inconveniente es el ancho de banda utilizado, en estas condiciones es importante que el video streaming permita la fragmentación del contenido del video, es recomendable que cada frame generado tenga diferente resolución y poder satisfacer eficientemente las características específicas de cada cliente [44]. El desarrollo digital propone avances en la comprensión y comunicación de video que pueden ser mostrados mediante el uso de dispositivos inalámbricos pero debido a su limitación de recursos especialmente de memoria, se necesita utilizar técnicas que permitan la visualización del video de forma eficiente. La técnica de video streaming cumple un rol importante en la constante búsqueda de calidad y rapidez de transmisión.

Video streaming ha experimentado un crecimiento exponencial, generándose muchos esfuerzos de investigación en relación a la codificación y transmisión de la información multimedia, con la finalidad de lograr video streaming eficiente, robusto, escalable y de baja latencia [45]. Las aplicaciones de video requieren gran ancho de banda, la calidad del video debe mantenerse en una red inalámbrica, la naturaleza impredecible del medio de transmisión tiene un efecto perjudicial en la calidad del video. Para cumplir con este objetivo es necesario una alta eficacia y eficiencia de compresión que puede lograrse utilizando CODEC que presenten: un bajo consumo de

energía, baja complejidad de codificación y decodificación y baja latencia [46] [47] [48] [49] [50].

Se ha dado un aumento de la demanda de aplicaciones multimedia debido al crecimiento de los servicios de comunicación, comercio y entretenimiento. Actualmente, los ordenadores portátiles y teléfonos presentan mayor capacidad de procesamiento que permite la transmisión de video más eficiente y eficaz. Las técnicas de codificación de video han causado gran impacto en la industria multimedia, ya que proporcionan secuencias de video de alta calidad utilizando el ancho de banda disponible; muy útiles en los servicios de video streaming móvil donde es significativo la eficiencia de almacenamiento y transmisión [51].

Con el video streaming nuevos modelos de negocio se están explorando 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... [52]. La tecnología inalámbrica también permitirá el incremento de casas y edificios inteligentes, autopistas automatizadas y redes en el cuerpo para el análisis y tratamiento de condiciones médicas, aplicaciones para la educación, salud… [53] [54].

Los proveedores de servicios de Internet han introducido redes inalámbricas en aeropuertos, hoteles, cafeterías..., y facilita proveer a bajo costo conexiones de Internet de altas velocidades para usuarios de teléfonos móviles. Aplicaciones tales como televisión digital, video bajo demanda, vigilancia de video doméstico y comercial, video llamadas son ampliamente populares. Uno de los objetivos de los proveedores de dichos servicios es entregar el video con alta calidad que satisfaga e incluso impresione al usuario. Este objetivo, lleva a la necesidad de la calidad del video.

En los próximos años se espera que video streaming sea la aplicación que domine el tráfico que se comparte en el Internet. Los usuarios podrían demandar resoluciones de imagen de alta calidad que pueden requerir anchos de banda más grande. Las estadísticas de uso, indican que hay demanda de videos en línea de larga duración como: noticias, deportes y entretenimiento [55] [56].

2.2.1. Protocolos y CODEC

El apogeo de la tecnología de video streaming ha sido posible gracias al desarrollo de los protocolos de comunicación y transporte, un protocolo es un conjunto de reglas establecidas entre dos dispositivos para permitir la comunicación entre ambos, los protocolos de comunicación utilizados en la transmisión de video streaming se muestran en la Figura 2.6.

Los protocolos de transporte se encargan del control de errores, la secuenciación y el control del flujo, en video streaming se utilizarán básicamente dos protocolos de transporte *Transmission Control Protocol* (*TCP*) y *User Datagram Protocol* (*UDP*), ideales para la difusión de contenidos multimedia.

En relación a los protocolos de comunicación, en la tabla 2.4 se presentan los protocolos de establecimiento y gestión de sesión por parte del cliente *Real Time Video Streaming Protocol* (*RTSP*), los relativos al transporte y control *Real Time Transport Protocol/ Real Time Control Protocol* (*RTP/RTCP*) con las principales funcionalidades.

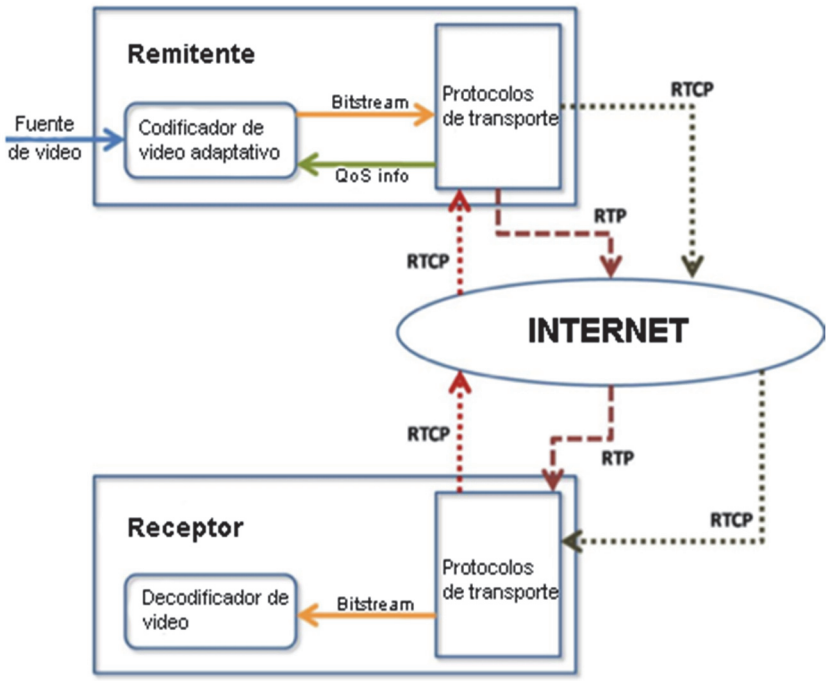

Figura 2.6 Protocolos utilizados en video streaming

Tabla 2.4 Protocolos de Comunicación

PROTOCOLO	DESCRIPCIÓN
RTSP	Establece y controla uno o muchos flujos sincronizados de datos, ya sean de audio o de video. RTSP es un protocolo no orientado a conexión, en lugar de esto el servidor mantiene una sesión asociada a un identificador, en la mayoría de los casos RTSP usa TCP para datos de control del reproductor y UDP para los datos de audio y vídeo. La conexión debe permanecer activa durante todo el proceso de comunicación, es decir, desde el momento en el que el cliente lanza la primera petición, hasta que termina de recibir los datos correspondientes al vídeo o audio y utiliza el propio protocolo RTSP para cerrar la sesión.
RTP	Es un protocolo que se utiliza para enviar cualquier tipo de formato de audio y video, funciona sobre el protocolo UDP, haciendo que el protocolo RTP no garantice que los paquetes lleguen a su destino y con la calidad de servicio ni en el orden de envío. Las aplicaciones que usan RTP son menos sensibles a la pérdida de paquetes, pero muy sensibles a los retardos, razón por la que se prefiere UDP sobre RTP. RTP además de los datos en sí, transporta cierta información útil para realizar un cierto control de la comunicación, pero no informa sobre el estado de la misma, es por esto que trabaja junto RTCP, el cual genera los mensajes de control de las transferencias que incluye información sobre los paquetes perdidos.
RTCP	Se utiliza para transmitir contenido multimedia en tiempo real y permite hacer un control directo en el cliente de la reproducción del contenido enviado desde el servidor mediante comandos como play y pause. Se basa en la transmisión periódica de paquetes de control durante una sesión, utilizando el mismo mecanismo de distribución de los paquetes RTP. La función principal de RTCP es informar de la calidad de servicio proporcionada por RTP. Este protocolo recoge estadísticas de la conexión e información adicional como: bytes enviados, paquetes enviados, paquetes perdidos entre otros. Una aplicación puede usar esta información para incrementar la calidad de servicio, ya sea limitando el flujo o usando un CODEC de compresión más baja.

Los códecs pueden codificar el flujo o la señal (para la transmisión, el almacenaje o el cifrado) y recuperarlo o descifrarlo del mismo modo para la reproducción o la manipulación en un formato más apropiado.

El CODEC puede servir en distintos estados de la información y con diferentes funcionalidades. Puede utilizarse en la transmisión de un archivo o en el almacenaje, para comprimir propiamente, para cifrar o para modificar su formato.

Existen CODEC que provocan pérdidas de información para conseguir un tamaño lo más pequeño posible del archivo destino, así como también existen CODEC sin pérdidas. Pero los más utilizados son los primeros, ya que no justifica un aumento de calidad casi imperceptible para el ser humano, por un notable aumento del tamaño de los datos. Los CODEC que no pierden información en el proceso de compresión, suelen emplearse cuando la calidad del archivo multimedia enviado por la Web, se requiere para un tratamiento posterior de dicho archivo.

El problema asociado al video digital es la gran cantidad de recursos de almacenamiento y ancho de banda que demanda, para solventar esta situación se han desarrollado varios estándares de compresión de video, los cuales eliminan la redundancia y permiten su transmisión de una forma compacta y eficiente. A esta labor se han enfocado dos organismos de estandarización que desarrollan estándares y algoritmos de compresión de video.

- La *International Telecommunication Union* (*ITU*) a través de los estándares "H", entre los que se destacan H.263 y H.264 relacionados con la codificación de video.
- La *International Organization for Standarization / International Electrotechnical Commission* (*ISO/IEC*) desarrolla los estándares conocidos como MPEG, entre los que se destacan MPEG 2 y MPEG 4.

Un CODEC de audio se basa en el algoritmo de compresión con pérdida, un proceso por el que se eliminan algunos de los datos de audio para poder obtener el mayor grado de compresión posible, resultando en un archivo de salida que suena lo más parecido posible al original.

2.2.2. Degradación de la calidad de experiencia del usuario

La percepción de calidad del usuario es importante en todo servicio que se le provea, hay que establecer los mecanismos e indicadores de calidad de experiencia en el servicio de video streaming a proporcionar al usuario. Los usuarios demandan calidad del video streaming el cual es muy dependiente del contenido.

La QoE toma en cuenta la percepción, expectativas y hábitos del usuario, que no solo depende de la valoración del rendimiento de la red, sino de enfocarse en evaluar el aspecto humano subjetivo [57]. La QoE es el criterio principal de diseño de video streaming y la gestión de recursos inalámbricos es una parte crítica en la prestación de servicio con calidad. De acuerdo a las predicciones de tráfico en la red, las dos terceras partes del total del tráfico móvil serían de video streaming en el 2017. Este enorme crecimiento de tráfico de video streaming en redes inalámbricas ocasiona enormes desafíos debido a la limitada disponibilidad de recursos inalámbricos en términos de frecuencia, tiempo y espacio, en adición a la naturaleza volátil de la tecnología inalámbrica. El problema de entregar QoE en redes inalámbricas ha sido abordado mediante avances tanto en video como en la tecnología inalámbrica. Esto incluye: eficiente compresión del video, técnicas inalámbricas avanzadas, video streaming adaptativo... [58].

Actualmente, video streaming es uno de los flujos de datos más consumidos en Internet, el cual demanda el mayor ancho de banda. Por lo tanto, video streaming tiene un impacto importante en la red global, esto confluye en lograr una buena calidad de experiencia del usuario, en donde el estado de la red no es el único parámetro considerado para la adaptación del video, sino que también son tomados en cuenta aspectos como el tamaño de la pantalla del dispositivo, CODEC utilizado y la velocidad de carga del buffer de datos durante la reproducción del video [59]. 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 [60]. El servicio de video streaming es percibido

por el usuario a través de un proceso complejo que implica el sistema visual y auditivo, combinando el sistema sensorial y el sistema cognitivo. En el aspecto sensorial se relaciona con el video y toma en cuenta aspectos como: brillo, color, forma y movimiento. En el aspecto auditivo se relaciona con: tono de audio, volumen y timbre [61] [62]. La QoE relacionada con las aplicaciones inalámbricas define factores humanos que influyen subjetivamente la percepción, por ejemplo: antecedentes culturales, expectativas personales, estado de emoción, así como la compensación multimedia y de contenido. La QoE para los servicios multimedia se relaciona fuertemente con la *Quality of Service* (*QoS*) (retraso, jitter y pérdida de paquetes), rendimiento del terminal y atributo de servicio. El tamaño del buffer de memoria es un factor clave entre el retraso y el jitter [63].

Entendiendo las necesidades importantes de la industria, ITU-T desarrollo modelos QoE estandarizados para predecir la calidad de audio, video y voz; el objetivo de estos modelos es predecir de forma subjetiva el *Mean Opinion Score* (*MOS*) [64] [65] [66].

En [67] se analiza los enfoques de medición de calidad de video propuestos en los últimos años:

- *Peak-Signal-to-Noise-Ratio* (*PSNR*): evalúa la diferencia de calidad entre fotografías [68] [69] [70].
- *Video Quality Metric* (*VQM*): desarrollado por el *Institute for Telecommunication Science* (*ITS*) que provee medidas objetivas para definir la calidad del video, mide los efectos de percepción de las deficiencias de video incluye: visión borrosa, ruido global, distorsión de color, distorsión de bloque.
- *Moving Pictures Quality Metric* (*MPQM*): métrica de la calidad de imagen en movimiento que incorpora dos características de visión humana: sensibilidad al contraste y enmascaramiento.
- *Structural Similarity Index* (*SSIM*): evalúa la calidad de video basado en la medición de distorsión estructural ya que la visión humana es altamente especializada en extraer información estructural.

- *Noise Quality Measure* (*NQM*): en esta se definen métricas como: medida de distorsión, calidad del ruido, variación en relación a la sensibilidad con la distancia, dimensiones de la imagen.

En [71] se define como métricas de calidad de experiencia a:

- *Buffering Radio* (*BR*): la fracción de sesión de video cuando el usuario se ha detenido, es el parámetro que presenta alta correlación con la no satisfacción de los usuarios terminando la sesión [72].
- *Low Bitrate Ratio* (*LBR*): el transcodificador en base al ancho de banda enviara los frames a una tasa de bits (alta o baja).

El uso de aplicaciones multimedia está aumentando, estas tendencias incluyen el desarrollo de la red inalámbrica que permite a los usuarios acceder a diferentes servicios. Lo más importante a ser tomado en cuenta en este tipo de redes es el retraso, colusión, atenuación y la interferencia. Hay varios problemas en las aplicaciones multimedia en tiempo real, la pérdida de paquetes y la caída de paquetes debido a la demora de tiempo excesiva, todo esto afecta a la calidad percibida por el usuario [73].

Otro factor que debe ser tomado en cuenta al proporcionar servicios de video streaming es que la QoE se influenciada por la QoS, ya que presenta parámetros que influyen en la percepción de calidad de los servicios por parte del usuario.

Es importante tratar de reducir el impacto negativo que ocasiona los fenómenos que afectan el rendimiento. La calidad de servicio de una red inalámbrica se mide cuantitativamente por varios aspectos del servicio de la red como: tasas de errores, latencia, ancho de banda, rendimiento, retraso en la transmisión, disponibilidad, latencia, pérdida de paquetes... [74] [75].

La QoS permiten 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) [76].

Para el video streaming, el retraso es un factor a considerar cuando se transmite el video al cliente. El máximo retraso en el audio de un video flujo que puede tolerar un usuario es 80 milisegundos. El nivel de pérdidas incurridas debido al retraso depende de la forma de transmisión de un flujo. Las aplicaciones de video streaming usualmente imponen un umbral o límite superior de pérdidas de paquetes. Es necesario, que la tasa de pérdida de paquete se mantenga bajo este umbral para lograr la calidad visual aceptable. El nivel de pérdida de paquetes es alta debido a la congestión de la red, esto causa degradación de la calidad de multimedia especialmente en redes inalámbricas con limitado ancho de banda [77] [78] [79].

La tecnología inalámbrica ha avanzado rápidamente proporcionando el aumento de ancho de banda a los usuarios; pero se hace necesario el uso inteligente y eficiente de los recursos inalámbricos para proporcionar una mejor QoE [80] [81] [82].

Los factores más importantes que provocan interrupciones en WiFi son:

- La banda 2.4GHz es utilizada por una gran cantidad de dispositivos de comunicación, incluso recibe interferencias de una gran cantidad de electrodomésticos, esto hace que la señal muchas veces se pierda, y sea casi imposible volver a reconectarse.
- Interferencias: sensible a emisiones de radio y de televisión, proximidad de otras redes inalámbricas, cámara inalámbrica.
- Factores atmosféricos: como la nieve, la lluvia o el granizo, pueden interferir en la señal.
- Limitación de ancho de banda.
- Tráfico de red.
- Pérdida de cobertura.
- Potencia de emisión.
- Protocolo de red inestable.

El desempeño de un servicio de video streaming puede ser notablemente afectado por la inconsistencia del canal, intermitencia en la conexión, bloqueos de la señal en el ambiente que introducen ruidos y ecos. También afecta los fenómenos que pueden

ocurrir en una transmisión mediante ondas de radio como son: la reflexión, difracción, absorción e interferencia, estos generan menor estabilidad en la conexión, disminución en el ancho de banda efectivo y mayores tasas de error.

Las condiciones del canal inalámbrico cambian en el tiempo que pueden ocurrir debido a interferencia y movilidad, convirtiendo al video streaming es una tarea difícil. Por lo tanto, es importante el aprovisionamiento de la calidad de servicio de extremo a extremo, necesario para el mantenimiento continuo de reproducción de video en las aplicaciones multimedia [83] [84].

Transmitir video streaming a través de redes inalámbricas es difícil porque este tipo de redes se expone a un ancho de banda variable en el tiempo, retraso, 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 [85] [86]. La velocidad de transmisión de los datos, es una medida de las prestaciones de la propia tecnología de transmisión, depende de numerosos factores como: efectos de atenuación, degradación de la señal, interferencias… [87]. Los medios de comunicación inalámbrica tienen baja fiabilidad con fluctuaciones de ancho de banda que conducen a la degradación de la calidad de video de manera significativa [88] [89].

El principal problema de video streaming inalámbrico es la fluctuación de su ancho de banda [90], diversidad de los terminales inalámbricos, ancho de banda y recursos limitados de los dispositivos inalámbricos [91] [92]. Los dispositivos inalámbricos han evolucionado vertiginosamente, lo que permite que nuevos servicios como video streaming sean posibles de realizarse. A pesar de este avance, la capacidad de memoria, batería y procesamiento son menores en relación a los computadores de escritorio, esto define la necesidad de generar aplicaciones que permitan el ahorro de las capacidades y lograr la calidad de experiencia en el usuario [93] [94] [95]. Actualmente las características de visualización y pantalla en un teléfono móvil se han desarrollado eficazmente, proveyendo de buenas experiencias en servicios video streaming, es importante cuidar este factor que afecta a la calidad de experiencia del

cliente. La continua movilidad del cliente, establece que pueda alcanzar zonas de no cobertura de la transmisión multimedia vía WiFi, provocando interrupciones en el servicio de video streaming [96]. Otros problemas como el roaming y el escaso control del usuario sobre las zonas de cobertura en las que se encuentra provocan desconexiones [97] [98]. Los operadores móviles están aprovechando WiFi para aliviar la presión de la demanda de ancho de banda creciente de aplicaciones. Sin embargo, los operadores suelen carecer de mecanismos inteligentes que controlen la forma que acceden los usuarios a las redes WiFi, esta falta de controles degrada la QoE. El aumento en el número de redes inalámbricas y dispositivos que se conectan a ellas está teniendo un efecto directo en la eficiencia de las conexiones inalámbricas. Los servicios de video streaming en dispositivos inalámbricos enfrentan retos como la necesidad de mantener la calidad de experiencia y servicio y ofrecer una operación adecuada en ambientes heterogéneos.

Una interrupción en el servicio de video streaming provocará la visualización de los fragmentos cargados en el buffer cliente, luego de un tiempo de espera, ocurrirá la pérdida de sesión. Al darse cuenta el usuario de la interrupción, probablemente solicite nuevamente el servicio, esto implica el desarrollo de nuevos retrasos: tiempo de solicitud del servicio, tiempo del servicio habilitado.

Cuando se produce una interrupción de larga duración la sesión se pierde y se debe reiniciar el video streaming, provocando la invocación de los parámetros de conexión y visualizando el video desde el inicio. Esto causa molestia en el usuario del teléfono móvil que termina por abandonar la sesión de video streaming. El principal problema de la interrupción de servicio es que el operador o gestor de contenidos multimedia pueden perder dinero debido al abandono del usuario, el canal inalámbrico se puede congestionar con frames de vídeo que nadie los verá y se puede degenerar el uso de otros servicios que en ese momento están usando la red inalámbrica. Cuando se dan interrupciones en el servicio video streaming afecta al distribuidor de contenidos, ya que el usuario no mantiene el nivel de fidelidad requerido, por el malestar y abandono de sesión que provoca un quiebre en la comunicación. Al producirse una interrupción el servidor de video streaming desconoce este evento, por lo que continuará transmitiendo los fragmentos del video, esto ocasiona congestión en el canal

inalámbrico. La congestión antes mencionada afecta a otros servicios que están utilizando el canal de comunicación. El desconocimiento por parte del servidor que se ha dado una interrupción, provoca el desperdicio de recursos del servidor. Video streaming en redes inalámbricas es un desafío que aún no ha sido solucionado eficientemente, existe dificultad de controlar las desconexiones intermitentes ocasionados por pérdida de cobertura u otras causas, 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 pérdida de efectividad de video streaming, ocasionando también pérdida de tiempo del usuario, afectando de forma directa en la QoE.

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 pueden producir desconexiones frecuentes del teléfono móvil. Estas desconexiones son totalmente impredecible, para mitigar su efecto adverso se requiere de una eficiente calidad de servicio y mecanismos de control de transmisión para adaptar a los cambios de la red [99] [100] [101].

2.3. JADE para control de interrupciones de video streaming

La técnica de agente es una de las más importantes tecnologías desarrolladas para apoyar las aplicaciones de Internet. Un 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 inteligentes representan unidades computacionales autónomas, que fundamentan su comportamiento en reglas ontológicas, 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 pro actividad, inteligencia y movilidad [102]. Otra definición describe los agentes inteligentes como entidades de software o hardware que exhiben características útiles tales como autonomía, modularidad, proactividad, adaptación, racionalidad, habilidad social, reactividad, continuidad

temporal, orientación hacia el objetivo final, movilidad, benevolencia, colaboración, entre otras [103].

Los agentes actúan de manera autónoma; están programados para que perciban y aprendan de su entorno, razonan sobre lo aprendido; eligen una o varias soluciones para resolver problemas o consultar requisitos automáticos o humanos; viajan sobre la plataforma de comunicaciones de una red; sirven para automatizar tareas que se ejecutan repetidamente; se programan una única vez y posteriormente no requieren manipulación humana. Un agente de software inteligente es un programa que tiene una acción sobre un comportamiento lo que implica la capacidad de decidir si la acción es apropiada. Los agentes pueden invocarse a sí mismos, para esto, el agente necesita de mucho conocimiento que lo apoye, la ontología es un buen camino para representar este conocimiento. El agente ejecuta una ontología si las acciones del entorno son consistentes con las definiciones de la ontología. Un agente es una entidad de software que puede actuar por sí mismo con el fin de alcanzar unos objetivos que se ha fijado inicialmente y además está caracterizado por una o varias cualidades tales como capacidad de razonamiento (inteligencia), percepción de su entorno y actuación con base a ciertas circunstancias (reactividad) [104]. También, tiene la capacidad de actuar en forma proactiva, es decir, actúa sin necesidad de darle una orden y la habilidad de desplazarse de un lugar a otro (movilidad), entre otras características. 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.

Los agentes presentan características como: autonomía, aprendizaje y proactividad; además un alto nivel de abstracción debido a su modularidad y sencillez en las comunicaciones. También su escalabilidad y flexibilidad son factores que apoya al desarrollo de aplicaciones distribuidas [105] [106]. Los agentes monitorean el progreso de las estrategias seleccionadas, se comunican intercambiando mensajes, explotan técnicas de resolución de problemas y también de aprendizaje basadas en inteligencia artificial, logrando un desempeño dinámico e independiente.

Las propiedades básicas de los agentes son: reactivo, proactivo, autónomo, orientado a objetos y tienen capacidad social. Los agentes tienen que establecer por si mismos su propia manera de aplicar recomendaciones y tomar una decisión de confianza. El uso de agentes proporciona mayor eficiencia y seguridad.

Un agente se desarrolla en un ambiente dinámico en el que se debe tomar en cuenta el tiempo de ejecución de acción y el estado inicial de un ambiente, ya que durante este tiempo el ambiente puede cambiar, por lo que el agente debe recolectar información del estado actual del ambiente, porque la meta lograda puede ser diferente a la esperada [107]. También debe tomar en cuenta la interferencia que otros procesos pueden producir sobre la acción a ejecutar, esto podría afectar el resultado esperado, por lo que debe coordinar y sincronizar las acciones con otros procesos o agentes.

El desarrollo orientado a agentes es paradigma de Ingeniería de software [108]. Su uso se ha intensificado en una gran variedad de aplicaciones, esto se debe a su autonomía y control sobre sus propias acciones y su estado interno; a la capacidad de analizar su entorno para responder de acuerdo los objetivos de su diseño. Otra razón es su flexibilidad ya que responde a los cambios que pueden ocurrir en el ambiente, direcciona su comportamiento y toma la iniciativa apropiada e interactúa con otros agentes, estos atributos alcanzados en una entidad de software ha influido en el desarrollo de este nuevo enfoque.

El concepto de agente constituye una potente herramienta de abstracción en el desarrollo de software, que facilita la construcción de sistemas distribuidos, inteligentes y robustos [109]. En la actualidad la tecnología de agentes inteligentes proporciona herramientas que permiten construir aplicaciones cuyas fuentes de información están distribuidas y que deben interactuar, en este entorno tecnológico donde nuevos y complejos servicios son demandados en los campos educativo, trabajo y ocio, el concepto de agente se consolida como una solución a las tendencias actuales: ubicuidad, interconexión, inteligencia, delegación y social.

Los agentes inteligentes móviles surgen como una tecnología para reducir el impacto de los problemas que presentan los dispositivos inalámbricos al ser partícipes de solución de aplicaciones personalizadas, reducir la latencia, controlar las

desconexiones de la red. Los agentes móviles pueden convertirse en un nuevo paradigma para programación distribuida de las redes de comunicación futuras.

El desarrollo tecnológico ha influenciado en la evolución de los sistemas de telecomunicación móvil, los entornos actuales convergen en el surgimiento de nuevos servicios cada vez más complejos como multimedia, video streaming, televisión digital. La tecnología de agentes ofrece muchas ventajas que pueden ser útiles en el diseño e implementación de servicios para dispositivos inalámbricos en ambientes inalámbricos ya que las características intrínsecas de los agentes ayudaran a solventar problemas de movilidad de usuarios, dispositivos heterogéneos y los problemas propios de la red.

El desarrollo de las tecnologías de la información y las comunicaciones en especial el avance de las comunicaciones inalámbricas ha permitido el auge del uso de dispositivos inalámbricos con mejores características en memoria, procesamiento, etc. Esto ha ocasionado que el usuario exija nuevos servicios como tv digital, video streaming, comunicación con sensores, entre otros. A pesar de las innovaciones realizadas en las características de los dispositivos inalámbricos éstos disponen de limitada capacidad de cálculo y no pueden manejar grandes cantidades de datos.

El inconveniente habitual de las comunicaciones inalámbricas es que al momento de utilizar el servicio, se genere una interrupción por varias razones entre ellas la falta de cobertura, puede ocasionar en el usuario la obligatoriedad de volver al punto de partida (inicio del video, descarga de un documento, llenar un formulario desde cero). Dentro del paradigma orientado a agentes inteligentes están cobrando especial importancia los agentes móviles, constituyéndose en programas que pueden parar en cualquier momento, trasladarse a otro dispositivo, continuar con la ejecución del servicio o tarea en el punto en el que se suspendió.

El uso de la programación orientada a agentes móviles permite crear aplicaciones, de monitorización de redes o de filtrado y búsqueda de información. Estas aplicaciones cobran especial relevancia al ser usadas en dispositivos inalámbricos con conexión a redes inalámbricas. Una de sus características principales propias del agente permite generar los cálculos o procesos en el dispositivo en el que se encuentran los datos ocasionado la disminución del tráfico de información en la red.

Para este tipo de sistemas es adecuado un sistema ontológico que proporciona una serie de ventajas:

- Sistema basado en el conocimiento: se utiliza ontologías y agentes con el fin de tener en cuenta el conocimiento previo recibido por expertos en una forma de reglas heurísticas, razonamiento basado en casos...
- Representación del conocimiento: el sistema de ontología se crea como un conjunto de ontologías del dominio correspondiente al ambiente, agentes y limitaciones.
- El sistema propuesto es inteligente y puede reaccionar adecuadamente frente a perturbaciones imprevistas, puede utilizar los resultados de la negociación entre agentes. Podría ser tanto reactivo y proactivo.

2.3.1. Ontología

El término ontológico se refiere a la formulación de un esquema conceptual riguroso de un domino específico, el cual facilita la comunicación y compartición de información entre diferentes sistemas, posibilitando que los recursos puedan accederse de acuerdo a su contenido semántico, priorizando la base de conocimiento y permitiendo la recuperación de la información de forma automática [110].

Las ontologías definen las conceptualizaciones de un dominio específico que deben ser modeladas y enriquecidas para que puedan ser entendidas por todos los agentes que forman parte de la arquitectura.

En el campo de la Inteligencia Artificial se la define como: *una especificación explícita de una conceptualización* [111] y en la disciplina de los Sistemas de Información se la considera como: *un artefacto del software (o lenguaje formal) diseñado para un conjunto específico de usos y ambientes computacionales* [112]. La Ontología es un esquema conceptual que permite especificar el conocimiento de uno o varios dominios que es compartido por varios agentes y es interpretable por un programa informático [113]. La ontología establece un vocabulario común necesario para que los agentes se comuniquen entre ellos, así la comunicación es facilitada a través una ontología compartida, el cual define los conceptos y las relaciones entre los conceptos de un

dominio en particular [114]. Hoy en día se entiende como ontología: *un esquema conceptual que permite especificar el conocimiento de uno o varios dominios que es compartido por varios agentes e interpretable por un programa informático*. Una ontología permite la representación del dominio con el objetivo de compartir, intercambiar y ser reutilizada por los agentes. La visión de una ontología para diferentes propósitos tales como aplicaciones Web, Web semántica, sistemas de información, está recibiendo mayor atención. Una ontología es una especificación de una conceptualización compartida, es un acuerdo para usar un vocabulario de forma consistente.

Pueden darse varios casos de video streaming donde los ambientes son distribuidos, heterogéneos y con tipos diferentes de sistemas. En estos casos se requiere de estructuras más flexibles y dinámicas, para ello es necesario la combinación del sistema basado en el conocimiento, el cual es presentado desde las ontologías y el uso de módulos inteligentes proactivos presentado por sistemas de agentes.

2.3.2. Sistema Multiagente

Un único agente no puede lograr la solución de problemas complejos, para aquello en lo que es necesario un conjunto de agentes que interactúan y buscan el logro de los objetivos comunes planteados en el sistema. El uso de varios agentes representa la descentralización de la aplicación y prioriza la comunicación como factor crítico de éxito.

Los sistemas existentes no tienen capacidad de colaboración, se utiliza agentes inteligentes porque cooperan entre ellos para realizar las actividades a ellos encomendadas. Un *Multiagent System* (*MAS*) es un sistema compuesto de múltiples agentes inteligentes interactuantes, estos son utilizados para resolver problemas difíciles de hacerlo por un agente individual [115] [116].

En un MAS, un conjunto de agentes inteligentes trabajan para logar un objetivo global, define un ambiente que contiene varios agentes, estos pueden interactuar uno con el otro a través de mensajes, en este entorno, el elemento fundamental es un

agente inteligente el cual tiene tres características como: habilidades sociales, proactivas y reactivas.

La programación orientada a agentes permite el desarrollo de MAS estructurados para interactuar y cooperar para resolver problemas que están fuera del alcance de cada agente, se destaca la implementación de estructuras cooperativas que son muy útiles para el control de video streaming en dispositivos inalámbricos [117].

Un MAS es aquel compuesto de varios agentes, los cuales desempeñan tareas encomendadas por un usuario u otro agente, deben ser autónomos es decir poseer las capacidad de decidir qué deben hacer para cumplir con sus objetivos de diseño e interactuar con otros agentes a través del intercambio de datos. Un MAS debido a que está constituido por varias entidades independientes (agentes) necesita definir muy bien cómo están organizados, cómo se comunican, coordinan y cooperan, y finalmente como se controlan estos agentes para lograr un objetivo común. Los agentes son caracterizados por diferentes visiones del Mundo que son explícitamente definidas por las ontologías, es decir, que las visiones que el agente reconoce son conceptos descritos por el dominio de la aplicación el cual está asociado al agente junto con sus relaciones y restricciones.

La interoperabilidad entre agentes se logra a través de la conciliación de los puntos de vista o visiones del mundo mediante el compromiso de ontologías comunes que permiten a los agentes interoperar y cooperar mientras mantienen su autonomía [118]. Han sido utilizados para construir aplicaciones distribuidas basadas en componentes. Existen una variedad de metodologías y herramientas que apoyan en el desarrollo de este tipo de sistemas, es ideal que se tomen en cuenta todos los aspectos del comportamiento de un agente y su despliegue. Es necesario entonces: 1) modelar los agentes en un sistema y las interfaces de estos agentes 2) describir la información consumida y generada por cada agente 3) esbozar las posibles interrelaciones entre agentes 4) 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.

Tabla 2.5 Sistemas MAS

PLATAFORMA	LICENCIA	LENGUAJE	DOMINIO
ABLE Agent Building and Learning Environment	Open Source	Able Rule Language	Construcción de agentes inteligentes haciendo uso de máquinas de aprendizaje y razonamiento
iGen The Cognitive Agent Software Toolkit	Propietario	C, C++, java	Modelado de diversos aspectos biológicos del ser humano
ADK Agent Development Kit	GPL	Java	Aplicaciones con una alta escalabilidad
ZEUS	Open Source	Visual Editors	Sistemas multi-agente basado en reglas y scripting
JASA Java Auction Simulator Application Programming Interface (API)	GPL	Java	Simulación de entornos económicos
AgentBuilder	Propietario	KQML, java, C++	Sistemas multi-agente de propósito general
JADE Java Agent Development Framework	GPL	Java	Sistemas multi-agente de propósito general

Es importante realizar el análisis de las plataformas que permiten el desarrollo de MAS y seleccionar la más idónea de acuerdo al escenario y el problema planteado. Como se puede revisar en la tabla 2.5 (adaptada de [119]) se han generado varias plataformas entre ellas JADE.

Cuando un grupo de agentes individuales conforman un MAS es necesario definir el lenguaje de comunicación que permita el intercambio de mensajes entre diferentes agentes. Se ha escogido el lenguaje de agentes de propósito general *Foundation for intelligent Physical Agent* (*FIPA*) *Agent Communication Language* (*ACL*) que se caracteriza por permitir que los agentes se comuniquen utilizando ontologías y lenguajes de contenido. FIPA es una organización de estándares de la IEEE que promueve la tecnología basada en agentes y la interoperabilidad de éstos con otras tecnologías. FIPA define todos los aspectos fundamentales de comunicación dentro de un MAS especificando protocolos de interacción, lenguajes de comunicación y ontologías. Establece referencias para la creación, registro, localización, comunicación y operaciones entre agentes. La comunicación entre agentes es fundamental para poder conseguir la potencia propia de los MAS. Para que los agentes se puedan comunicar deben usar el mismo lenguaje de comunicación. El lenguaje de comunicación de agentes

(FIPA ACL) permitirá transmitir una serie de conocimiento que vendrá expresado en un lenguaje de contenido [120].

La tecnología de agentes se ha convertido en un nuevo campo en las ciencias de la computación, este avance ha generado un buen número de plataformas de diferentes calidades y madurez. En [119]se evalúa las plataformas ASDK, Ajanta, Tryllian`s ADK, FIPA-OS, Grasshopper, JADE, JACK Intelligent Agent, Zeus, de acuerdo a parámetros como: compatibilidad, comunicación, movilidad, políticas de seguridad, usabilidad; parámetros seleccionados de acuerdo a los estándares definidos en las recomendaciones FIPA para implantar plataformas de agentes.

En el análisis JADE ocupa el segundo lugar luego de Grasshopper, ya que se caracteriza por ser open source, tener buena documentación, muy buena interfaz gráfica, aceptación de los usuarios y es utilizado en muchos proyectos de desarrollo. Perfeccionado en base a FIPA, cuenta con buenas características de seguridad, varios protocolos de comunicación. 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 [121].

2.3.3. JADE

Fue desarrollado en Italia conjuntamente por *Centro Studi e Laboratori Telecommunicazioni* (*CSELT*). Es un middleware completamente implementado en Java, que simplifica la implementación de sistemas multiagente ya que proporciona un conjunto de herramientas graficas que permiten la depuración en las fases de desarrollo y despliegue [122]. Incluye: 1) un entorno de ejecución donde los agentes JADE pueden residir y deben estar previamente activos en un host determinado, para que uno o más agentes puedan invocarlos, 2) una biblioteca de clases que se puede utilizar para el desarrollo de agentes y 3) un conjunto de herramientas graficas que permite la administración y supervisión de la actividad de los agentes activos [123]. Proporciona tanto un entorno de desarrollo como un entorno de ejecución para la realización y

mantenimiento de sistemas multiagente, proporciona un conjunto de herramientas gráficas que permiten tareas de depuración, monitorización, documentación y soporte. JADE se basa en estándar FIPA adoptado por la mayoría de plataformas MAS.

Las comunicaciones de agentes FIPA se especifican formalmente en el lenguaje de comunicación ACL para codificación, semántica y uso de los mensajes que indica como interaccionan y son gestionados los agentes. El mensaje FIPA se define por la ontología en la que se describen los términos del contenido del mensaje.

Algunas de las características que JADE ofrece son:

- Es un middleware ya que permite el funcionamiento de agentes sobre plataformas heterogéneas.
- Ejecución de cada agente como un hilo separado, capaz de ejecutarse en máquinas remotas.
- Provee servicios de páginas amarillas y blancas.
- Proporciona API para crear, suspender, reanudar, bloquear, despertar, mitigar, clonar y destruir agentes.
- Apoya a la movilidad de agentes entre procesos y diferentes máquinas.
- Provee de herramientas para analizar los mensajes de comunicación entre agentes.
- Soporte para ontologías y lenguajes de contenido.
- Integración con tecnologías Web.
- Proporciona aplicaciones gráficas que facilita la monitorización y depuración.

Cada instancia del entorno de ejecución de JADE se llama contenedor. Los agentes viven en los contenedores, que son procesos Java provistos por JADE en tiempo de ejecución y proporcionan todos los servicios necesarios para la ejecución de los agentes. Hay un contenedor especial, llamado contenedor principal que representan el punto de arranque de la plataforma: es el primer contenedor que sería lanzado y el resto de contenedores deben unirse a este contenedor mediante el registro correspondiente.

JADE posibilita la implementación de agentes independiente de la plataforma en la que se va a ejecutar, está conformada por uno o más contenedores que pueden estar

ubicados en diferentes hosts, los agentes JADE pueden comunicarse con plataformas desarrolladas con otras tecnologías y que residen en contenedores distribuidos; el conjunto de todos los contenedores se denomina plataforma. En cada plataforma debe existir un contenedor principal que es el primero en ejecutarse y en el que se registrarán el resto de contenedores que buscan comenzar su ejecución.

JADE de acuerdo a la Figura 2.7, está compuesto por un *Agent Management System* (*AMS*), *Directory Facilitator* (*DF*) y *Agent Communication Channel* (*ACC*). AMS es un agente que supervisa el control sobre el acceso y uso de la plataforma de los agentes. Solo un AMS existe en una plataforma, ofrece el servicio de páginas blancas y ciclo de vida del servicio, mantiene un directorio de identificadores de agentes y el estado del agente. Cada agente debe registrarse en un AMS con la finalidad de obtener su identificación válida. DF es un agente que provee el servicio de páginas amarillas en la plataforma. ACC es el componente de software que soporta todo el intercambio de mensajes dentro de la plataforma, incluyendo mensajes desde y hacia las plataformas remotas [124].

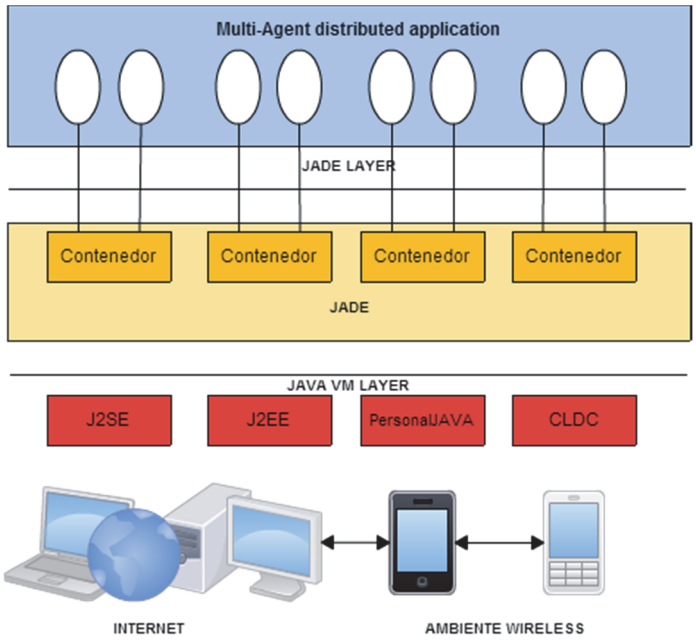

Figura 2.7 Arquitectura JADE

JADE permite el manejo de ontologías para representar el dominio de aplicación mediante conceptos, predicados, acciones, tipos de agentes, etc. También posibilita el intercambio de elementos de la ontología entre agentes.

Una ontología en JADE es una instancia de la clase *jade.content.onto.Ontology*, en el cual los esquemas han sido adicionados para definir los tipos de predicados, acciones de los agentes y los conceptos relevantes del dominio abordado. Por lo tanto, es posible declarar los slots definiendo la estructura: tipo de predicado, acción del agente y el concepto. Cada esquema incluido en una ontología debe asociarse con una clase de Java o interface. Esto proporciona una capa robusta en la cual la comunicación orientada a objetos entre agentes puede facilitarse ya que el mecanismo de ontologías automáticamente llena los objetos esquema con datos basados en reglas y restricciones de la ontología definida.

Los agentes no solamente deben hablar un mismo idioma, sino que también deben tener una ontología en común. Las ontologías son formas de representar el conocimiento, capturan el conocimiento acerca de un dominio de interés. Una ontología describe el concepto en un dominio y también la relación que tienen entre conceptos como por ejemplo: es sub-tipo de, es una instancia de... Explotar el lenguaje de contenido y la ontología incluida en JADE, permite a los agentes hablar y razonar acerca de “cosas y hechos” relacionados a un dominio dado, debe seguirse los siguientes pasos: 1) definir una ontología incluyendo los esquemas para los tipos de predicados, las acciones de los agentes y concepto pertinente al dominio. 2) desarrollar clases propias de Java para todos los tipos de predicado, acción del agente y concepto en la ontología 3) Seleccionar un lenguaje adecuado entre los directorios soportados por JADE 4) registrar la ontología definida y seleccionar el lenguaje de contenido para el agente 5) crear y manejar expresiones de contenido como objetos Java, que son instancias de las clases desarrolladas y permitir que JADE traduzca estos objetos desde/hasta strings o secuencias de bytes que se ajustan a la ranura de contenido del mensaje ACL.

Para posibilitar que JADE pueda ejecutarse en dispositivos inalámbricos se desarrolló JADE LEAP. Utiliza el servicio de MAS que permite el diseño de la mensajería y *Remote Method Invocation* (*RMI*) [125]. LEAP, que se puede desplegar sobre J2ME

mediante un modo partido (Split). Esto significa que el contenedor donde se encuentra el agente se divide en: FrontEnd, ubicado en el dispositivo móvil, y un BackEnd en el servidor remoto, los cuales siempre están comunicados, logrando la distribución de la carga computacional [126] [127].

JADE ayuda al manejo de desconexiones intermitentes mediante su protocolo *Message Transport Protocol* (*MTP*) 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 y se negocie la reproducción del video desde el punto en que se generó la interrupción.

2.4. Trabajos relacionados y estado del arte

El mecanismo de proxy y uso de buffer ha sido utilizado por varios autores para reducir el ancho de banda requerido en las redes inalámbricas, en [128] aplica un algoritmo que permite dividir al video en diferentes segmentos y graba el video de forma parcial en la cache, definiendo las compensaciones entre el buffer del cliente, la necesidad de almacenamiento en el proxy y la velocidad del video transmitido en la red; también el proxy es utilizado a través de una política de grabación mediante el cual el video es grabado en diferentes niveles correspondiente al tamaño de la información y al ancho de banda requerido; esta política permite al proxy ajustar la cantidad de cada video basado en la condición del recurso y el patrón de acceso del usuario [129]; otra aplicación del proxy es su localización 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, el cual permite al proxy determinar los paquetes que deben ser enviados desde el servidor de medios o retransmitidos directamente desde el proxy, buscando reducir la tasa de distorsión del video streaming [130] y por último, el desarrollo de un esquema que permite lograr una calidad en la transmisión a través del estudio del problema de la distancia existente entre el proxy y el cliente [131] .

Se ha utilizado la arquitectura de proxies para adaptar la velocidad de transmisión, teniendo en cuenta el estado del canal inalámbrico, en [132] 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.

Se han generado varias propuestas que aplican la arquitectura de proxies y recursos previamente almacenados, orientados principalmente a mitigar los efectos de la latencia, pérdida de paquetes, demoras… y mejorar la continuidad de transmisión de video streaming, en [133] se propone una técnica de almacenamiento en cache que despliega proxies multimedia a lo largo del camino del servidor al cliente, en esta técnica un servidor proxy almacena los frames iniciales de videos más demandados, ante un solicitud del cliente, el servidor proxy inicia la transmisión de los frames iniciales y a la vez solicita los frames restantes al servidor del video, 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. También se presenta una arquitectura de distribución de video asistida por un proxy [134] que reduce los requisitos exigidos al servidor y la experiencia de mejora de la latencia en el cliente, para ello presenta técnicas de selección de videos populares y asignación inteligente de recursos, las cuales son asistidas por el proxy y proveen servicio instantáneo a un gran número de clientes.

La arquitectura de proxies se implanta para controlar interrupciones de video streaming en redes inalámbricas y proporcionar la continuidad del servicio en dispositivos móviles; en [135] se define un middleware para video streaming móvil, embebido en el sistema operativo, que mediante la monitorización de eventos, garantiza la integridad en una interrupción en ambientes inalámbricos; sin embargo el usuario para mantener la continuidad del servicio debe seleccionar los archivos locales cache provistos por el middleware y funciona en una tecnología especifica de celulares; otros autores proponen un mecanismo de sincronización de flujos en ambientes móviles mediante un proxy utiliza el método de almacenamiento en el buffer para gestionar y controlar el despliegue del video en el cliente, en este se almacenan los frames del

servidor y se encarga del envío a los terminales móviles; manteniendo un flujo continuo del video streaming mediante un esquema de sincronización, al ajustar la duración del frame lo que provee una velocidad de reproducción constante incluso en frecuentes desconexiones de corta duración, esto posibilita la continuidad del flujo cuando ocurre una interrupción; bajo este enfoque, se propone un middleware, que proporciona servicio de tolerancia a desconexiones a los dispositivos móviles en ambientes inalámbricos basados en la intensidad de la señal de la red [136], aplicando esquema de memorización intermedia desarrollado en Java para dispositivos inalámbricos Android y por último, en [137] se propone una estrategia de buferización para mantener la continuidad del servicio de video streaming independiente de la movilidad del cliente para dispositivos inalámbricos.

Un valor añadido a las propuestas anteriores la presentan otros autores al implementar los proxies como agentes de software para gestionar interrupciones de video streaming en dispositivos móviles y proporcionar continuidad del servicio. En [138] se presenta una plataforma propietaria para resolver las desconexiones que mantiene la continuidad del servicio de video streaming personalizado aplicando un esquema de buffers proactivo y adaptativo que precargan los contenidos de multimedia; otros autores han desarrollado un middleware con agentes [139], que gestiona la provisión del contenido multimedia a los dispositivos clientes en redes inalámbricas, 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 donde las sesiones personalizadas de readecuación proactiva permiten mantener la continuidad del servicio. Por último, en [140] se presenta un midleware basado en agentes para distribuir video bajo demanda, los agentes se comportan como proxies, capaces de negociar el nivel de calidad y el flujo de frames 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.

Existen trabajos, desarrollados por los integrantes de nuestro grupo de investigación, los cuales presentan propuestas más cercanas a resolver el problema planteado, en [141] se propone un protocolo basado en la arquitectura de proxies, que implementa la técnica de gestión de buffer para controlar interrupciones en la comunicación inalámbrica multimedia y recuperar la sesión de video streaming automáticamente, el mecanismo desarrollado se aplicó a móviles de plataforma específica Nokia; otros autores [142] proponen un mecanismo de control basado en la gestión de buffers proactivos de datos multimedia manejados por agentes JADE LEAP, que se encargan de resolver las desconexiones intermitentes WiFi lograr la reanudación automática de las sesiones RTSP en dispositivos inalámbricos, mecanismo aplicado a una gama de celulares Nokia.

Se pueden establecer varias deficiencias en los trabajos anteriores:

- Se generaron soluciones para plataformas específicas, ya que la arquitectura de hardware heterogéneo que presentaban los dispositivos móviles así como el software de programación propietario imponían rígidas restricciones al desarrollar aplicativos para ellos.
- El uso de Web sockets reales no es posible porque éstos no manejan conexiones de datos usando RTP solo comandos de control mediante el protocolo TCP.
- Los agentes inteligentes desarrollados, no utilizan ontologías solo se establece una comunicación sencilla entre ellos mediante el protocolo *Message Transport Protocol* (*MTP*).

Nuestro interés es posibilitar la continuidad del video streaming en dispositivos inalámbricos, tomando en cuenta el comportamiento impredecible de la comunicación inalámbrica y las características propias del dispositivo; las propuestas aplican modelos de patrones software y nuevas tendencias como son servicios Web y agentes, para resolver la reanudación automática del video al suscitarse una interrupción.

2.5. Contribuciones a la mitigación de las interrupciones de video streaming inalámbrico

Nosotros proponemos varias soluciones a la mitigación de las interrupciones del servicio de video streaming en terminales inalámbricos. En ninguno de los trabajos antes relacionados hemos encontrado que se analice la producción de software cliente servidor para el servicio de video streaming. Nosotros hemos encontrado los patrones software adecuados para este diseño, proponiendo una arquitectura novedosa en la que basamos nuestras soluciones de mitigación de las interrupciones del servicio de video streaming en redes inalámbricas.

2.5.1. Patrones software de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y entonces se describe el núcleo de la solución a dicho problema de tal forma que se pueda usar esta solución un millón de veces si hacer dos veces de la misma forma.

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.

De acuerdo a [143] buen patrón debe cumplir los siguientes requisitos:

- Debe solucionar un problema.
- Son conceptos probados no teorías.
- La solución no es obvia, genera una solución para un problema indirectamente.
- Describe una relación es decir describe sistemas, estructuras o mecanismos más profundos.

- Debe tener un componente humano, recurrir explícitamente a la estética y utilidad.

A continuación se describe los patrones software utilizados en el presente trabajo de investigación, tomado de [144]:

Tabla 2.6 Componentes MVC

VISTA	CONTROLADOR	MODELO
En la vista se produce la visualización de las interfaces de usuario. Permite al usuario interactuar con la aplicación. Se trabaja con los datos pero no se realiza un acceso directo a los mismos. Se compone de la información que se envía al cliente y los mecanismos de interacción con éste.	Es el enlace entre las vistas y los modelos. Se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. Realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario.	Representa la información con la que trabaja la aplicación. Donde se definen mecanismos para: • acceder a la información • actualizar el estado Contiene una representación de los datos que maneja el sistema, su lógica de negocio relacionada con los datos, y sus mecanismos de persistencia.

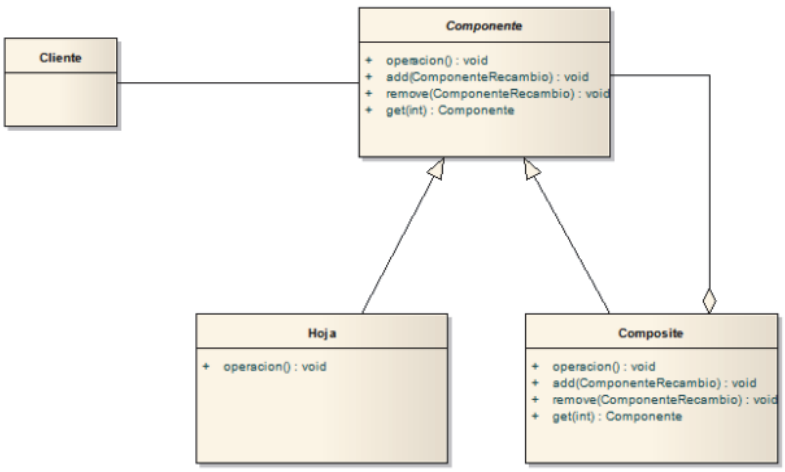

Figura 2.8 Patrón Composite

- *Patrón Modelo Vista Controlador* (*MVC*): separa en componentes distintos los datos, la interfaz de usuario y la lógica de control, promulga orden, reutilización, flexibilidad… los componentes de este patrón y su detalle se puede observar en la tabla 2.6.
- *Patrón Composite*: compone objetos en estructura de árbol para representar jerarquías de todo parte, permite que los clientes traten de manera uniforme a los objetos individuales. En base a la Figura 2.8, se enfoca en la construcción de objetos complejos a partir de objetos simples y similares entre sí, utilizando composición recursiva y definiendo la estructura jerárquica de elementos anidados. El objetivo de este patrón es la facilidad de uso y apoya a la vista del patrón MVC.
- *Patrón Strategy*: define una familia de algoritmos, encapsula a cada uno de ellos y los hace intercambiables. De acuerdo a la Figura 2.9, este patrón permite que los algoritmos o estrategias prestablecidas varíen de forma independiente de los clientes que lo usan. Su importancia radica en que encapsula un algoritmo sin preocuparse de los detalles de implementación y permite un cambio de estrategia en tiempo de ejecución. El objetivo de este patrón es lograr mayor nivel de fiabilidad y flexibilidad. Apoya al controlador en el patrón MVC.

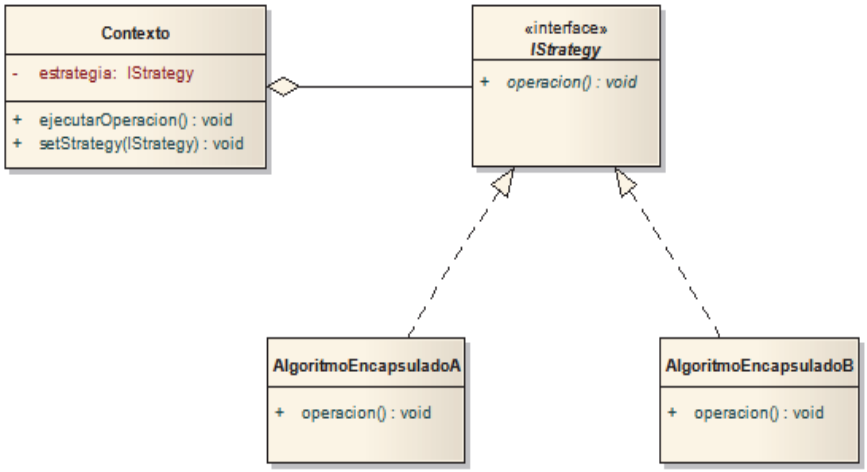

Figura 2.9 Patrón Strategy

- *Patrón Observer*: define una dependencia de uno a muchos entre objetos de forma que, cuando un objeto cambia de estado, se notifica a los objetos dependientes para que se actualicen automáticamente. En este patrón diferentes objetos suscriptores están interesados en el cambio de evento de un objeto observado y reaccionaran de forma independiente cuando este cambio ocurra, puede observarse en la Figura 2.10. El objetivo de este patrón es mantener bajo acoplamiento. Apoya al modelo en el patrón MVC.
- *Patrón Proxy*: proporciona un sustituto o intermediario para otro objeto de modo que pueda controlarse el acceso que se tiene hacia él. De acuerdo a la Figura 2.11, es una entidad a la que se delega la ejecución de procesos, y decidirá qué actividades realizar para su consecución. Se tienen varios tipos de proxies.
    - Proxy remoto: se encarga de la comunicación entre el cliente y el objeto remoto.
    - Proxy virtual: coordina la instanciación de objetos que realizan operaciones computacionales costosas únicamente cuando el acceso al objeto es requerido.
    - Proxy de protección: controla el acceso a un objeto en base a reglas de autorización.

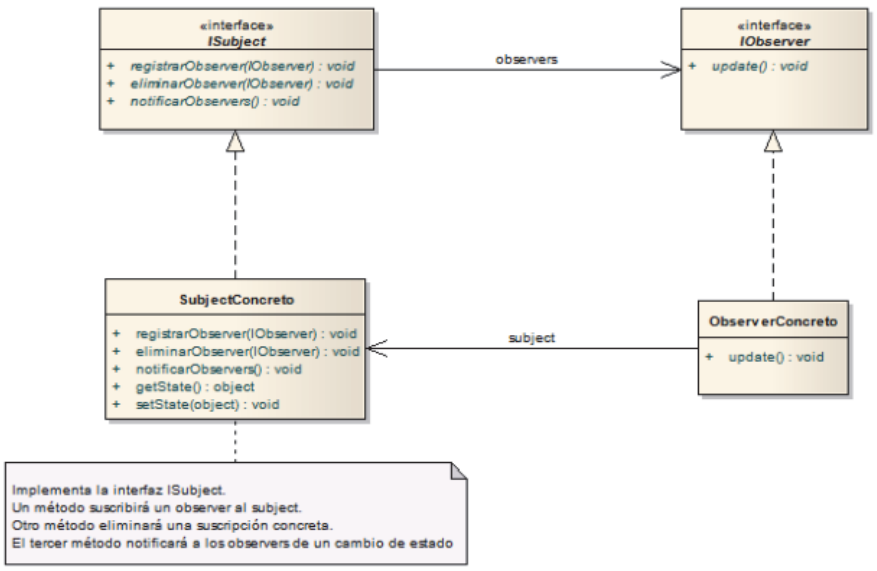

Figura 2.10 Patrón Observer

- *Patrón Adapter*: convierte la interfaz de una clase en otra interfaz que el cliente espera. De acuerdo la Figura 2.12, define una capa intermedia que permite comunicarse a dos clases con interfaces incompatibles, a través de una adaptación de la interfaz. Este patrón se aplica cuando se necesita transformar una estructura a otra, pero sin modificar la clase original.

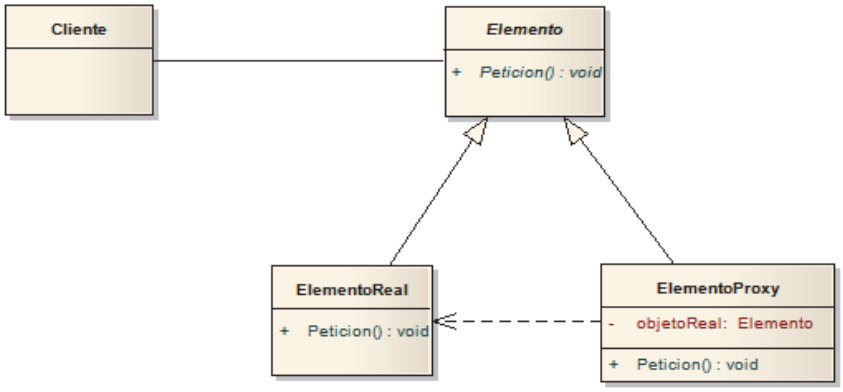

Figura 2.11 Patrón Proxy

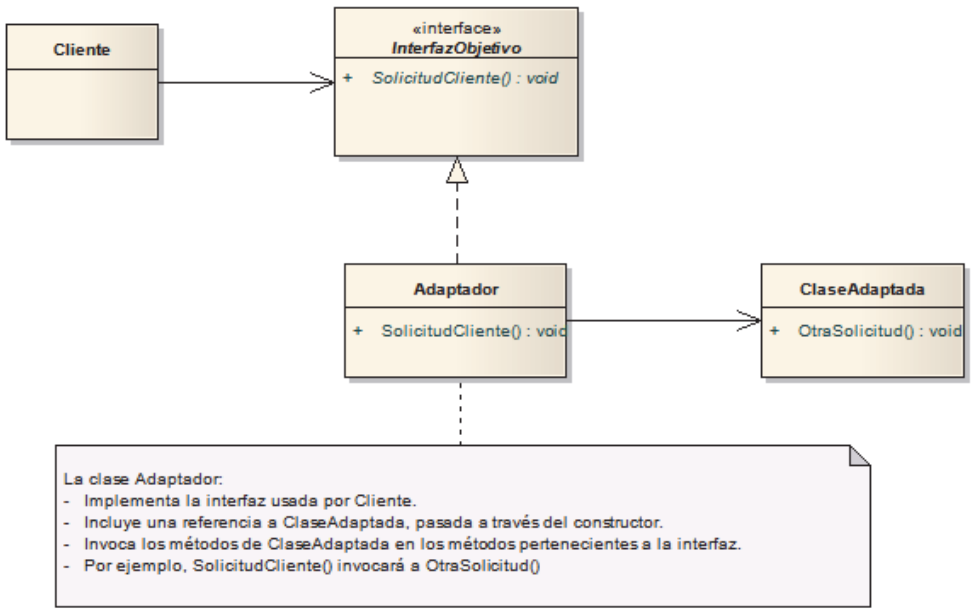

Figura 2.12 Patrón Adapter

2.5.2. JADE-WSIG

Además de usar el modelado de patrones software para diseñar el servicio básico de video streaming inalámbrico, también hemos usado servicios Web basados en add ons de la plataforma JADE (JADE-WSIG) que proporciona interoperabilidad y nos permite diseñar una solución multiplataforma. Hasta donde alcanza nuestro conocimiento esta es la primera vez que se utiliza esta plataforma para este propósito.

Los servicios Web [145] [146] [147] propician la integración de aplicaciones que han sido desarrolladas en diferentes lenguajes de programación y se ejecutan sobre distintas plataformas. Constituyen un conjunto de funcionalidad accesible por el protocolo Simple Object *Access Protocol* (*SOAP*) [148] [149] basado en mensajes *eXtensibility Markup Language* (*XML*) [150] [151].

Para que el servicio Web pueda ser leído e interpretado se utiliza *Web Services Description Language* (*WSDL*) [152] [153], que es un contrato entre el proveedor y el cliente, en éste se establecen los detalles de transporte de mensajes y su contenido. La descripción concreta del servicio se puede publicar en un servicio de registro como *Universal Description*, *Discovery and Integration* (*UDDI*) [154], frecuentemente descrito como el servicio de páginas amarillas de los servicios Web.

Los servicios Web son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web, las cuales intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario.

Un servicio Web es un componente al que se puede acceder mediante protocolos Web estándar, utilizando XML para el intercambio de información. Un servicio Web es una colección de métodos a los que se puede llamar desde cualquier lugar de Internet, siendo este mecanismo de invocación totalmente independiente de

la plataforma que se utilice y del lenguaje de programación en el que se haya implementado internamente el servicio.

Los servicios Web ofrecen información con un formato estándar que puede ser entendido fácilmente por una aplicación, son componentes de aplicaciones distribuidas que están disponibles de forma externa, se pueden utilizar para integrar aplicaciones escritas en diferentes lenguajes y que se ejecutan en plataformas diferentes. Los servicios Web son independientes de lenguaje y de la plataforma gracias a que los vendedores han admitido estándares comunes.

El *World Wide Web Consortium* (*WC3*) define un servicio Web como un sistema software diseñado para soportar interacciones máquina a máquina a través de la red. Dicho de otro modo, los servicios Web proporcionan una forma estándar de interoperación entre aplicaciones software que se ejecutan en diferentes plataformas. Por lo tanto, su principal característica su gran interoperabilidad y extensibilidad así como por proporcionar información fácilmente procesable por las máquinas gracias al uso de XML. Los servicios Web pueden combinarse con muy bajo acoplamiento para conseguir la realización de operaciones complejas. De esta forma, las aplicaciones que proporcionan servicios simples pueden interactuar con otras para "entregar" servicios sofisticados.

Un Servicio Web está formado por los siguientes componentes:

- *Lógica*: se trata del componente que procesa la petición para generar la información solicitada por el cliente. Básicamente resuelve el “problema” y puede, para ello, comunicarse con otros servicios Web, acceder a bases de datos o bien invocar API de otras aplicaciones solicitando la información (o parte de ella) que ha de generar para enviar en formato XML.
- *SOAP*: protocolo de comunicación, basado en XML, que sirve para la invocación de los Servicios Web a través de un protocolo de transporte, como *HyperText Transfer Protocol* (*HTTP*). Consta de tres partes: una descripción del contenido del mensaje, unas reglas para la codificación de los tipos de datos en XML y una representación de las llamadas *Remote Procedure Call*

(*RPC*) para la invocación y respuestas generadas por el Servicio Web. El mensaje SOAP está compuesto por un envelope (sobre), cuya estructura está formada por los siguientes elementos: header (cabecera) y body (cuerpo).

- *UDDI*: directorio donde es posible publicar los servicios Web, permitiendo con ello que los posibles usuarios de ese servicio puedan obtener toda la información necesaria para la invocación y ejecución del servicio Web. Un directorio UDDI ofrece una serie de interfaces que posibilitan tanto la publicación como la obtención de información sobre los servicios Web publicados.
- *WSDL*: lenguaje basado en XML que permite la descripción de los Servicios Web definiendo la gramática que se debe usar para permitir su descripción y capacidades (datos, comandos que aceptan o producen), y su publicación en un directorio UDDI.

Los servicios Web deben caracterizarse por lo siguiente:

- *Comunicación ubicua*: la conexión de cualquier sistema o dispositivo a Internet debe garantizar la disponibilidad para cualquier otro sistema o dispositivo conectado a Internet.
- *Formato de datos universal*: cualquier sistema compatible con estándares abiertos como mensajes de texto autodescriptivos puede comprender y compartir los servicios Web XML y permitir la comunicación entre sistemas autónomos y heterogéneos.
- *Interoperabilidad*: un servicio debe permitir su utilización por clientes de otras plataformas.
- *Amigabilidad con Internet*: la solución debe poder funcionar para soportar clientes que accedan a los servicios desde Internet.
- *Interfaces claramente definidas*: no debería haber ambigüedad acerca del tipo de dato enviado y recibido desde un servicio.
- *Aprovechar los estándares de Internet existentes*: en la implementación del servicio Web y evitar reinventar soluciones a problemas que ya se han resuelto.

- *Soporte para cualquier lenguaje*: el servicio Web es independientemente del lenguaje de programación en el que se halla escrito el cliente.
- *Distribuida*: la solución no debe estar ligada solo a una infraestructura de componentes en particular.

El middleware JADE [155], proporciona add-ons que permiten lograr la accesibilidad y colaboración en aplicativos Web. WSIG es un software add-on para JADE que posibilita que los agentes expongan sus operaciones como servicios Web, proporcionando interconectividad y escalabilidad. WSIG soporta el estándar de servicios Web que consiste de WSDL para descripciones de servicios, SOAP para transporte de mensajes y el repositorio UDDI para publicar los servicios Web.

WSIG promulga el uso de las ontologías que no solo incorporan datos sino contenido semántico, mediante la descripción de conceptos y sus relaciones para que formen parte del conocimiento de un agente y permitan la comunicación entre agentes al compartir mismo idioma y vocabulario, JADE WSIG posibilita que los agentes expongan sus operaciones como servicios Web, proporcionando interconectividad y escalabilidad [15]. Soporta el estándar de Web Services que consiste de WSDL para descripciones de servicios, SOAP para transporte de mensajes y el repositorio UDDI para publicar los Web Services, contiene dos elementos, mostrados en la Figura 2.13:

- *WSIG Servlet*: es el front – end hacia el mundo del Internet y responsable de atender las solicitudes HTTP/SOAP, y preparar la acción que es enviada al Agente WSIG, una vez que la acción ha sido ejecutada se encarga de convertir el resultado obtenido en un mensaje SOAP y generar una respuesta que es entregada al cliente solicitante del servicio.
- *WSIG Agent*: es el puente entre la Web y el mundo de agentes y es responsable de direccionar las acciones recibidas del WSIG Servlet al agente correspondiente para que ejecute el servicio.

Los agentes JADE publican sus servicios en el DF mediante una estructura llamada DF-Agent-Description basada en la especificación FIPA donde se detalla los servicios que

provee el agente, las ontologías a usarse para acceder al servicio y las acciones que se ejecutarán. Dichas descripciones son mapeadas a un WSDL y publicadas en un registro UDDI. WSIG requiere de JADE 3,5 o superior para ser instalado, siendo WSIG una aplicación Web es necesario tener un Contenedor Servlet que puede ser Glassfish o Apache Tomcat.

En resumen, WSIG se encarga de conectar la plataforma JADE con los servicios Web, provee de mecanismos de transformación que aseguran la integración, sin necesidad de cambiar las especificaciones de JADE. Su función es de ser un intermediario, cuando un agente quiere comunicarse con un servicio Web, WSIG se encarga de administrar y transformar este pedido para posteriormente enviarlos a los servicios Web correspondientes. Al retornar desde el servicio Web los resultados, estos son receptados por WSIG para traducir esta respuesta y reenviarla al agente correspondiente.

WSIG proporciona una arquitectura de referencia estándar que hace posible la interoperabilidad y extensibilidad con diferentes aplicaciones y permite su combinación para ejecutar los servicios solicitados.

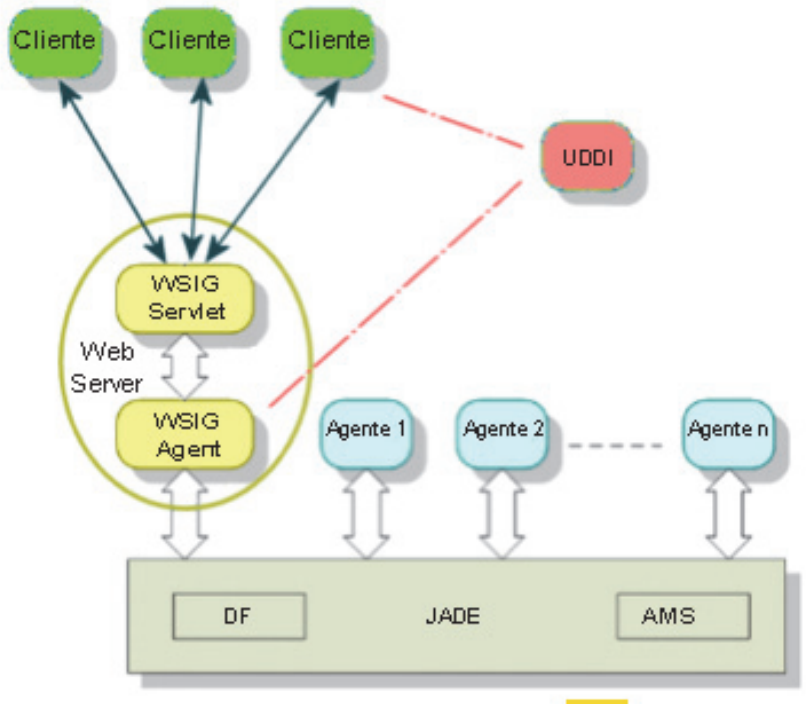

Figura 2.13 Componentes de WSIG

2.5.3. Esquema básico cliente servidor de video streaming con patrones

Para poder entender las soluciones al problema de video streaming que exponemos en los siguientes capítulos, en este apartado presentamos el diseño de base del servicio de video streaming que iteramos en los siguientes capítulos.

Estamos interesados en analizar la afectación de las interrupciones al tiempo total de ejecución, la cantidad de retransmisiones que puede ocasionar y el impacto de una interrupción en la QoE del usuario, para ello se analiza el mecanismo del video streaming en su conceptualización básica que no presenta control de interrupciones.

Como primer paso en el planteamiento propuesto, se realiza el análisis de requisitos funcionales utilizando los modelos de casos de uso y su descripción (Figura 2.14). Se puede distinguir dos casos de uso, el primero abarca todas las actividades tendientes a iniciar la solicitud del video, negociar con el *Proxy* y determinar la autorización de acceso al servidor de video streaming; el segundo contiene el proceso y las actividades que buscan el envío de los frames a un buffer temporal y el despliegue de los mismos en el cliente.

En la tabla 2.7 se expone el caso de uso solicitar video que engloba las actividades a cumplirse desde la petición de servicio por parte del cliente, donde el *Proxy* cumple la labor de intermediario para validar el pedido y comunicárselo al servidor.

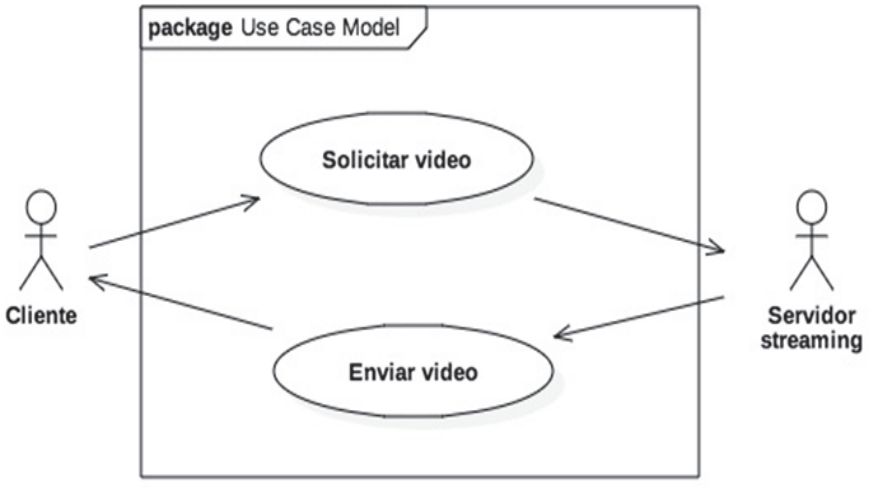

Figura 2.14 Caso de uso de esquema básico

En la tabla 2.8 se establece el detalle del caso de uso enviar video, inicia con el servidor de video streaming quien determina si el pedido es de video almacenado o tiempo real lo que define la fuente de origen del servicio solicitado, los frames se graban en el buffer y despliegan en la ventana del usuario.

Tabla 2.7 Caso de uso solicitar video

Caso de Uso: **solicitar video**
Actores: cliente, servidor video streaming
Resumen: se describe puntualmente como inicia la petición del video
Precondiciones:
Descripción: Curso Normal: 1. El cliente solicita la visualización del video 2. Ingresa los datos del puerto y servidor 3. Los datos son validados por el *Proxy* 4. Al no existir problemas el *Proxy* solicita el servicio al servidor de video streaming
Observaciones:
Poscondiciones:

Tabla 2.8 Caso de uso enviar video

Caso de Uso: **enviar video**
Actores: servidor video streaming, cliente
Resumen: se describe puntualmente como se realiza la entrega del video al cliente
Precondiciones:
Descripción: Curso video almacenado: 1.1. El servidor toma la información del almacenamiento de videos Curso video tiempo real: 1.2. El servidor toma la información capturada por la cámara 2. Se despliega una ventana en la que inicia el video streaming 3. El servidor inicia la entrega periódica de los flujos al almacenamiento temporal
Observaciones:
Poscondiciones:

Para plasmar este mecanismo básico de video streaming en patrones software, se ha seleccionado el patrón MVC, ya que promulga la organización del código, permite una mejor estructura y orden, centraliza el cuidado de los datos propiciando la integridad de estos, administra el dinamismo al cambio en sus componentes y fomenta la reutilización de componentes. Este diseño puede observarse en la Figura 2.15.

El detalle de cada componente del MVC se indica a continuación:

- *Modelo*: se encuentra definido por el almacenamiento temporal donde se alojan los flujos que son desplegados por el servidor de video streaming.
- *Vista*: se ubican las clases que van a permitir la visualización del servicio del video streaming en el cliente, organizando el contexto de reproducción en base a las características del dispositivo cliente.
- *Controlador*: se encarga de receptar los pedidos de video streaming del dispositivo cliente, analizar la información y coordina que el servidor recepte el pedido y se despliegue el video solicitado.

Con miras a implantar el modelo basado en el patrón MVC se define el diagrama de clases, Figura 2.16. Asociando las clases Reproductor Video y URL Streaming a la capa vista; la clase Coordinador de Servicio y Admisión se encuentran en la capa controladora y en la capa modelo tenemos las clases Almacenamiento Bajo Demanda, Video Tiempo Real y Buffer.

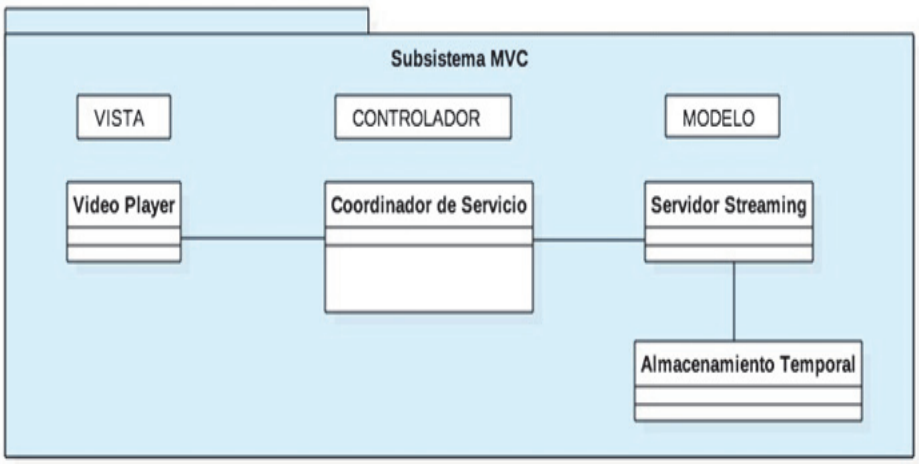

Figura 2.15 Esquema básico basado en patrones software

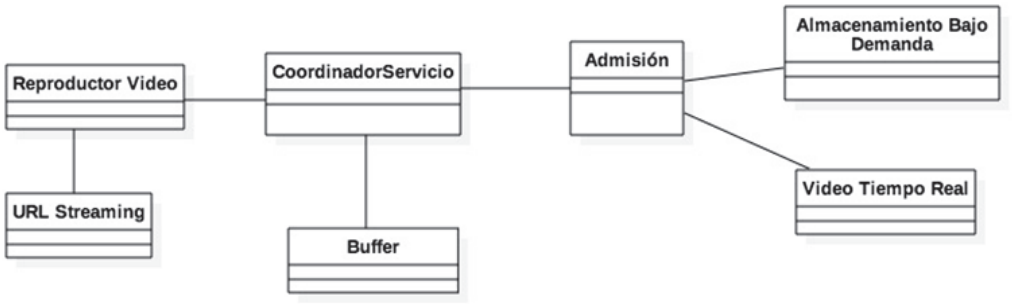

Figura 2.16 Diagrama de clases del esquema básico

En la Figura 2.17 se expone el diagrama de secuencia que indica la interacción del cliente al solicitar el video mediante la clase *Reproductor de Video*. Para esta solicitud es necesario definir el parámetro *URL Streaming* con el que se eleva la petición al *Coordinador de Servicio*. Esta petición puede tener dos tipos de video: video bajo demanda y en tiempo real, cada uno de ellos alimentados por fuentes diferentes, es decir, video bajo demanda lo obtiene de una librería de videos y video en tiempo real de un dispositivo que permita capturar el video (cámara, tarjetas de captura…).

El *Coordinador de Servicio* pide parámetros de sesión al servidor. Este entrega al *Coordinador de Servicio*, que a su vez entrega al *Reproductor de Video*. Una vez establecida la sesión de video streaming el reproductor solicita el video al servidor. Este envía el video al *Coordinador de Servicio*, quien establece el flujo de datos con el *Buffer* del cliente. Finalmente, cuando el *Buffer* recibe la información, se encarga de entregarlo al *Reproductor del Video* para su visualización.

En la Figura 2.18 se expone el diagrama de despliegue, donde el cliente se encuentra en cualquier dispositivo móvil y contiene: *Reproductor de Video*, *URL Streaming* y *Buffer*. El servidor se aloja en un servidor de aplicaciones que contiene: *Admisión*, *Almacenamiento bajo demanda* y *Video tiempo real*. Es importante resaltar que el *Coordinador de Servicio* que representa el *Proxy*, se ubica en la capa controladora y se encuentra distribuido entre el Cliente y el Servidor para atender sus peticiones y respuestas a las solicitudes.

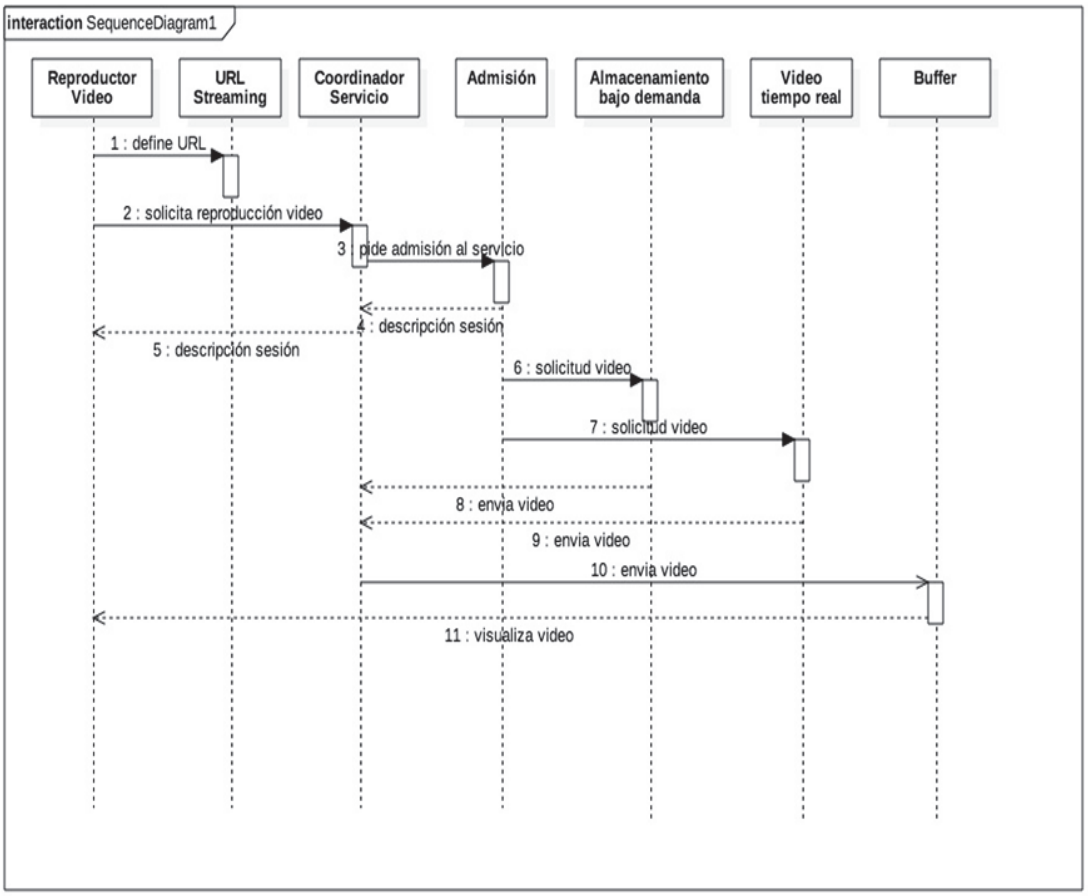

Figura 2.17 Diagrama de secuencia del esquema básico

Basado en el diagrama de secuencia, explicado en el apartado anterior, el *Reproductor de Video* envía el mensaje interno para instanciar la *URL Streaming*. Con esta información el *Reproductor de Video* envía una petición RTSP para solicitar parámetros de servicio al servidor. El servidor de video streaming mediante el protocolo *Session Description Protocol* (*SDP*) entrega la descripción de la sesión al cliente. Una vez establecida la sesión el *Coordinador del Servicio* solicita mediante mensaje interno enviar el video al cliente de alguna de las dos fuentes especificadas en la *URL Streaming*.

El servidor entrega de la fuente elegida (video bajo demanda, video en tiempo real) al *Coordinador de Servicio* el flujo de datos mediante mensaje interno. El *Coordinador del Servicio* envía este flujo de datos recibido a través del protocolo RTP al *Buffer* del cliente. El *Buffer* entrega al *Reproductor de Video* el flujo de datos para su visualización mediante mensaje interno.

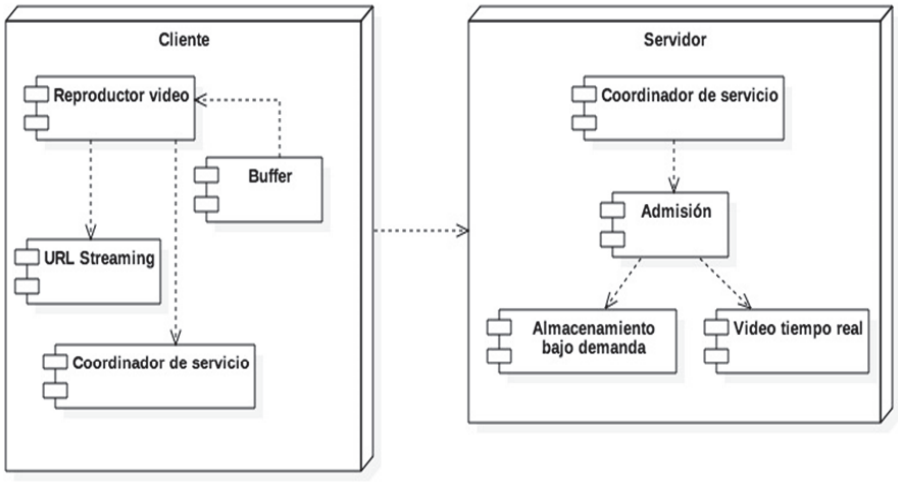

Figura 2.18 Diagrama de secuencia del esquema básico

2.5.4. Modelo matemático de rendimiento del esquema básico

A partir del modelo de despliegue anterior somos capaces de analizar un modelo sencillo de rendimiento de la implantación del esquema cliente servidor de base de video streaming, con el objetivo de poder comparar, en los siguientes capítulos, las ventajas que supone mitigar las interrupciones del servicio de video streaming con nuestras soluciones.

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

Si analizamos que el tiempo de ejecución el T_{exec} está determinado por la siguiente ecuación:

$$T_{exec} = T_{start_1} + T_{start_2} + T_{start_3} + \cdots + T_{start_n} +$$

$$T_{data_1} + T_{data_2} + T_{data_3} + \cdots + T_{data_n} +$$

$$T_{int_1} + T_{int_2} + T_{int_3} + \cdots + T_{int_n}$$

Donde:

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

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

T_{int} 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 *Tstart* presenta el mismo tiempo de duración, por lo que:

$$T_{start} = T_{start_1} += T_{start_2} = T_{start_3} = \cdots = T_{start_n}$$

Entonces:

$$\begin{aligned} T_{exec} = \quad & (n+1) * T_{start} \\ & + T_{data_1} + T_{data_2} + T_{data_3} + \cdots + T_{data_n} \\ & + T_{int_1} + T_{int_2} + T_{int_3} + \cdots + T_{int_n} \end{aligned}$$

Luego si agrupamos los *Tdata* transmitidos más de una vez, agrupamos los *Tint* presentados y separamos el $Tdata_n$ transmitido una sola vez, tenemos:

$$\begin{aligned} T_{exec} = (n+1) * T_{start} \\ + (T_{data_1} + T_{data_2} + T_{data_3} + \cdots) + T_{data_n} \\ + (T_{int_1} + T_{int_2} + T_{int_3} + \cdots + T_{int_n}) \end{aligned}$$

Si aplicamos el sumatorio tenemos la ecuación 1:

$$T_{exec} = (n+1) \times T_{start} + \sum_{i=1}^{n} (T_{data_i}) + \sum_{i=1}^{n} T_{int_i} + T_{data_n} \qquad (1)$$

$Tdata_n$ es la sumatoria de todos los flujos que han sido retransmitidos más el flujo que ha sido enviado una sola vez, se determina por la siguiente ecuación.

$$T_{data_n} = (T_{data_1} + T_{data_2} + T_{data_3} + \cdots + T_{data_{n-1}}) + T'_{data_n}$$

Aplicando la sumatoria tenemos:

$$T_{data_n} = \sum_{i=1}^{n} \left(T_{data_i}\right) + T'_{data_n}$$

Reemplazando *Tdata*$_n$ en la ecuación 1 se obtiene:

$$T_{exec} = (n+1) \times T_{start} \quad + \sum_{i=1}^{n} \left(T_{data_i}\right) + \sum_{i=1}^{n} T_{int_i} + \sum_{i=1}^{n} \left(T_{data_i}\right) + T'_{data_n}$$

Obteniéndose la ecuación definitiva:

$$T_{exec} = (n+1) \times T_{start} \quad + 2\sum_{i=1}^{n} \left(T_{data_i}\right) + \sum_{i=1}^{n} T_{int_i} + T'_{data_n}$$

Donde:

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

$\sum_{i=1}^{n} \left(T_{data_i}\right)$ es el tiempo total de *Tdata* transmitidos más de una vez.

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

T'_{data_n} es el tiempo del dato transmitido una sola vez.

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.

En la Figura 2.19 puede observase el comportamiento de los flujos transmitidos frente a una interrupción, en el peor escenario.

Se analiza a cada flujo transmitido para establecer el número de retransmisiones que el flujo puede tener al ocurrir una interrupción.

$$T_{flujo_1} * \frac{LongitudFrame}{capacidadCanal} = n$$

$$T_{flujo_2} * \frac{LongitudFrame}{capacidadCanal} = n - 1$$

....

$$T_{flujo_{n-1}} * \frac{LongitudFrame}{capacidadCanal} = 1$$

$$T_{flujo_n} * \frac{LongitudFrame}{capacidadCanal} = 0$$

Obteniéndose la siguiente ecuación (2):

$$Número\ de\ retransmisiones = n - i + 1 \qquad (2)$$

Donde *i*: es el número secuencial de ocurrencia de la interrupción.

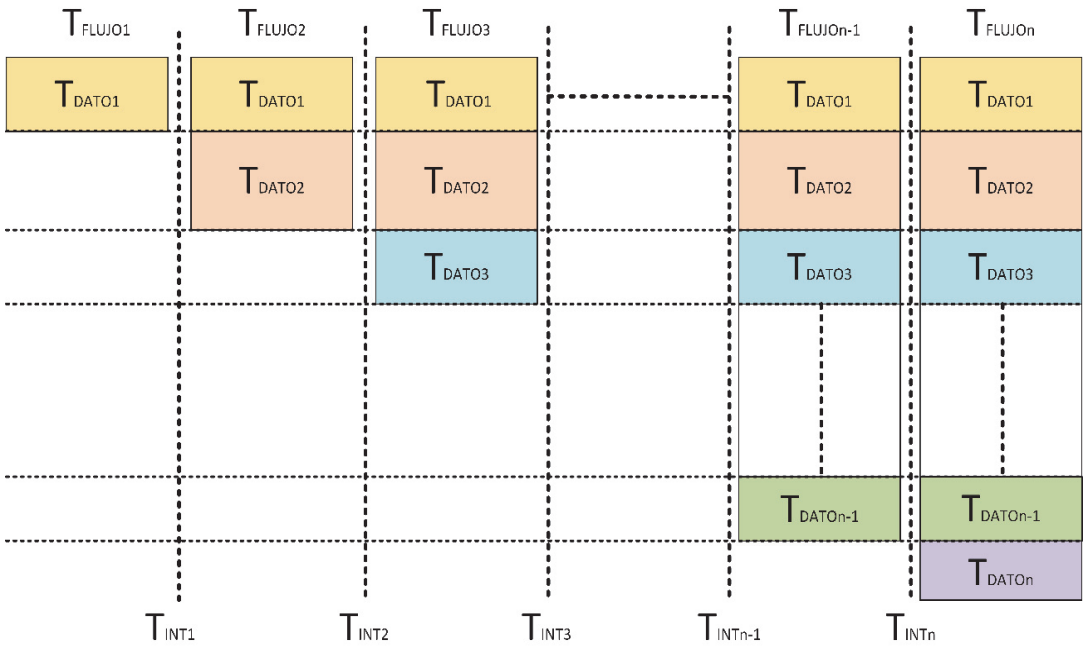

Figura 2.19 Análisis peor escenario

Para obtener en este peor escenario la cantidad de frames que no se retransmiten se muestra en la Figura 2.19 el $Tflujo_n$ que es la transmisión de todo el video, es igual a:

$$T_{flujo_n} = T_{dato_1} + T_{dato_2} + \ldots + T_{dato_n}$$

El $Tflujo_{n-1}$ que es el flujo transmitido antes de la última interrupción, de esto se deriva el número de frames que no se retransmiten con la siguiente ecuación.

$$Numero\ de\ frames\ no\ retransmitidos = T_{flujo_n} - T_{flujo_{n-1}}$$

Concluyéndose que el número frames no retransmitidos sería siempre mucho menor, en este escenario ya que las interrupciones ocurren cuando el usuario ha visto la mayor cantidad del video.

En la Figura 2.20 puede observase el comportamiento de los flujos transmitidos frente a una interrupción.

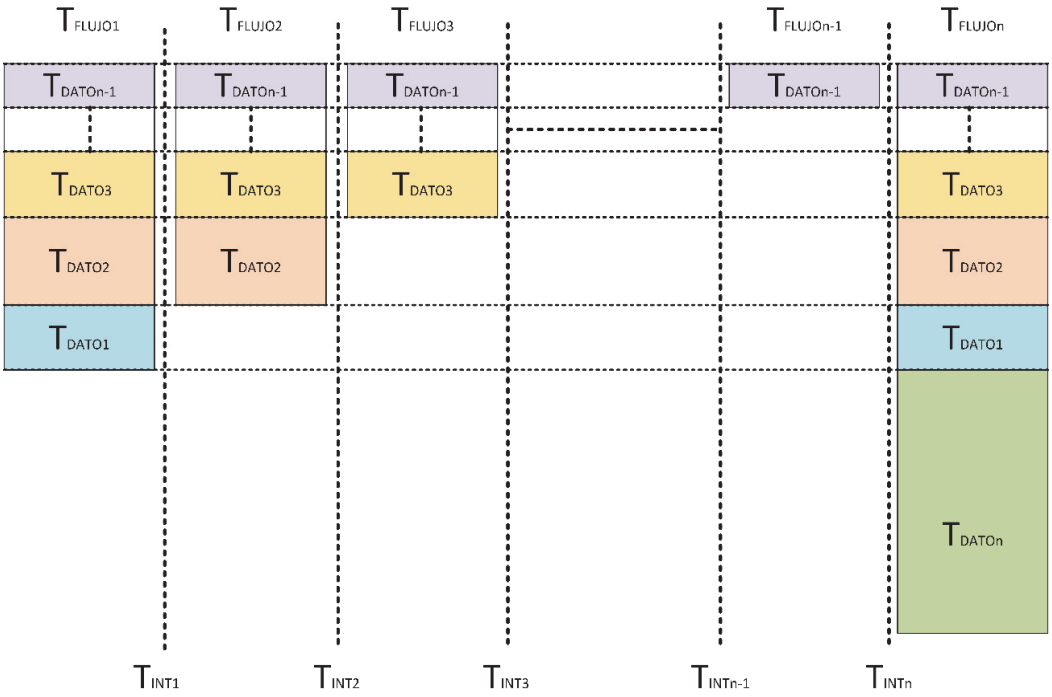

Figura 2.20 Análisis mejor escenario

Se analiza a cada flujo transmitido para establecer el número de retransmisiones que éste puede tener al ocurrir una interrupción.

$$T_{flujo_1} * \frac{LongitudFrame}{capacidadCanal} = 1$$

$$T_{flujo_2} * \frac{LongitudFrame}{capacidadCanal} = n - 1$$

….

….

$$T_{flujo_{n-1}} * \frac{LongitudFrame}{capacidadCanal} = n$$

$$T_{flujo_n} * \frac{LongitudFrame}{capacidadCanal} = 0$$

Obteniéndose la siguiente ecuación:

$$Número\ de\ retransmisiones = n - i + 1$$

Donde *i*: es el número secuencial de ocurrencia de la interrupción.

Para establecer en este mejor escenario la cantidad de frames que no se retransmiten, se observa en la Figura 2 el *Tflujon* que es la transmisión de todo el video que es igual a:

$$T_{flujo_n} = T_{dato_1} + T_{dato_2} + \ldots + T_{dato_n}$$

El *Tflujo1* que es el flujo transmitido con la primera interrupción, que representa el flujo con mayor cantidad de frames, con lo que se establece la siguiente ecuación.

$$Numero\ de\ frames\ no\ retransmitidos = T_{flujo_n} - T_{flujo_1}$$

Concluyéndose que este número sería siempre mucho mayor ya que las interrupciones ocurren en los inicios de la visualización del video. Realizando una comparación ente los dos escenarios se puede establecer que en el peor escenario, la cantidad de retransmisión de los frames ocasiona una congestión en el uso de la red inaceptable.

También es importante indicar que el tiempo de retransmisión de frames siempre es menor en el mejor escenario en comparación con el peor escenario.

Con la finalidad de evaluar el modelo matemático es necesario utilizar una nueva métrica que es el número veces que un flujo de bytes es retransmitido en base al número de interrupciones dadas. Para determinar el tiempo invertido en repetir los flujos cuando ocurre una interrupción, se plantean las siguientes condiciones ideales:

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

Es necesario determinar la cantidad de bits transmitidos en *Tdata*, la siguiente ecuación permite calcular obtener el número de bits.

$$Número\ de\ bits = Tdata * capacidadCanal$$

Si el canal es *Asymmetric Digital Subscriber Line* (*ADSL*) con WiFi transmite a 10 Mbps y si *Tdata1* tuvo un tiempo de ejecución de 16,20 segundos, entonces la ecuación para obtener el número de bits que se transmitieron seria:

$$Número\ de\ bits = 16{,}20 * 10\ = 162$$

Para obtener la cantidad en bytes lo dividimos para 8 obteniéndose 20,25 MB.

Para definir el análisis de la cantidad de bytes que son retransmitidos en cada flujo se procede a designar un valor a cada *tdata* para transformarlo a número de bytes y poder analizar el efecto de las retransmisiones, se determina 3 interrupciones y se obtiene los tiempos de $Tdata_1$ = 28 s, $Tdata_2$= 4 s, $Tdata_3$ = 12 s y $Tdata_n$ = 32 s; aplicando la ecuación anterior se obtiene la cantidad de bytes que se han transmitido en cada uno de ellos: $Tval_1$ = 35 MB, $Tval_2$ = 5 MB , $Tval_3$ = 15 y $Tval_n$ = 40 MB. En la Figura 2.21 estos valores son ordenados de menor a mayor para poder establecer de forma gráfica la cantidad de veces que son retransmitidos cada flujo, también puede analizarse que hay flujos que no se retransmiten.

Tdata2	Tdata3	Tdata1	Tdatan	# retrans
4 s	12 s	28 s	32 s	
5	5	5	5	3
	10	10	10	2
		20	20	1
			5	
Tval2	Tval3	Tval1	Tvaln	
5 MB	15 MB	35 MB	40 MB	

Figura 2.21 Flujos retransmitidos

Tdata	Tamaño bytes	bytes retransmitidos	# retrans	total bytes retrans
28,00	35,00	20,00	1	20
12,00	15,00	10,00	2	20
4,00	5,00	5,00	3	15
Total video s	Total video MB	Total retrans MB	%	
32	40,00	55,00	138	

Figura 2.22 Total de flujos retransmitidos

En la Figura 2.22 se define matemáticamente cada tdata y su correspondiente tamaño en bytes, también se detalla la cantidad de bytes de cada flujo que se retransmiten y el número de retransmisiones en total que presenta cada flujo por efecto de las interrupciones, para este número se aplicó la ecuación 2. Tambien puede observarse el total de video transmitido y el total de retransmisión, el 138% es totalmente un factor negativo influyente en la QoE del usuario.

La Figura 2.23 indica la relación entre el número de veces que los flujos han sido retransmitidos, en este escenario el flujo de menor tamaño se retransmite por más ocasiones, a pesar de ser un flujo de menor tamaño en bytes la cantidad total de bytes retransmitidos afecta directamente al servicio de video streaming que enfrenta interrupciones, provocando degradación de calidad percibida por el cliente, quien desea recibir un servicio continuo y efectivo.

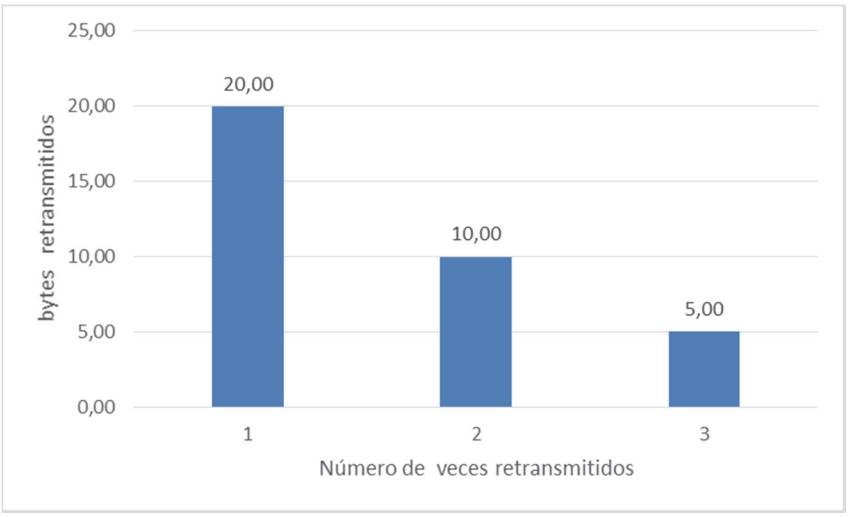

Figura 2.23 Análisis de retransmisión de flujos

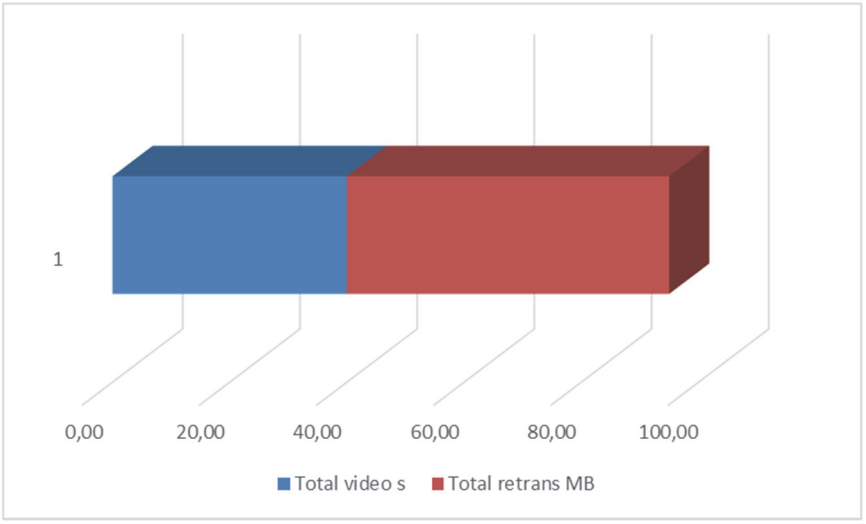

Figura 2.24 Efectos de la retransmisión

La cantidad de bytes adicionales que deberá soportar el cliente al darse interrupciones puede observarse en la Figura 2.24, el efecto que ocasiona las retransmisiones superan negativamente a los niveles soportados por el usuario.

Para corroborar los resultados obtenidos en el supuesto caso planteado anteriormente, se ha procedido a realizar una experimentación real en el mejor escenario, generando interrupciones al inicio del video. En la Figura 2.25 puede observarse todos los datos obtenidos tanto de la transmisión como de la interrupcion y el tiempo total de ejecución.

	Tstart	Tdata1	Tint1	Tstart	Tdata2	Tint2	Tstart	Tdata3	Tint3	Tstart	Tdatan'	Tdatan	a	b	c	Texec
													(n+1) * Tstart	∑(Tdatai)	∑(Tinti)	a+ 2(b) + c + Tdatan'
s	3,93	16,20	12,21	3,93	22,18	10,90	3,93	11,80	16,06	3,93	1168,82	1219,00	15,72	50,18	39,17	1324,07
MB	4,91	20,25	15,26	4,91	27,73	13,63	4,91	14,75	20,08	4,91	1461,03	1523,75	19,65	62,73	48,96	1655,09

Figura 2.25 Experimentación mejor escenario

Tdata	Tamaño bytes	bytes retransmitidos	# retrans	total bytes retrans
22,18	27,73	7,48	1	7,48
16,20	20,25	5,50	2	11,00
11,80	14,75	14,75	3	44,25

Total videos	Total video MB	Total retrans MB	%
1219	1523,75	62,73	4

Figura 2.26 Total de retransmisión mejor escenario

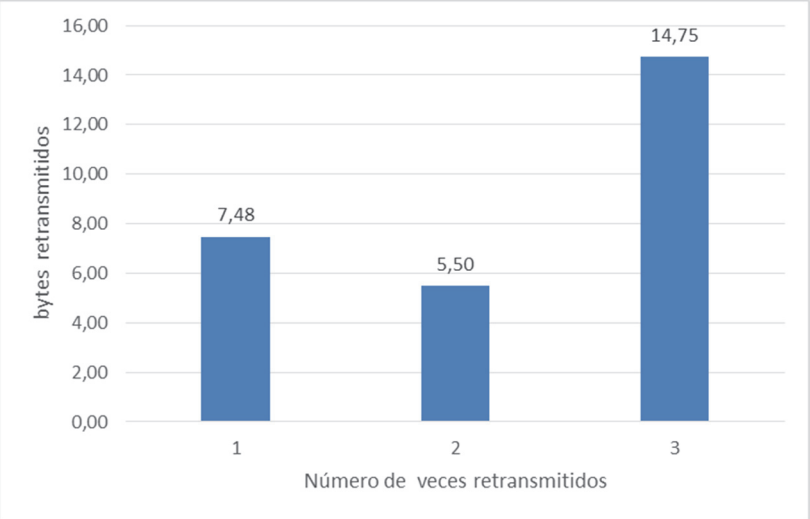

Figura 2.27 Resumen de retransmisiones mejor escenario

La definición de los datos, su transformación en bytes y la cantidad de retransmisiones de cada flujo se han establecido en la Figura 2.26. En este mejor escenario, se puede observar el total de video transmitido y la cantidad de bytes retransmitidos, definiéndose que se da un menor porcentaje de afectación cuando el usuario ha visualizado pocos frames y enfrenta una interrupción.

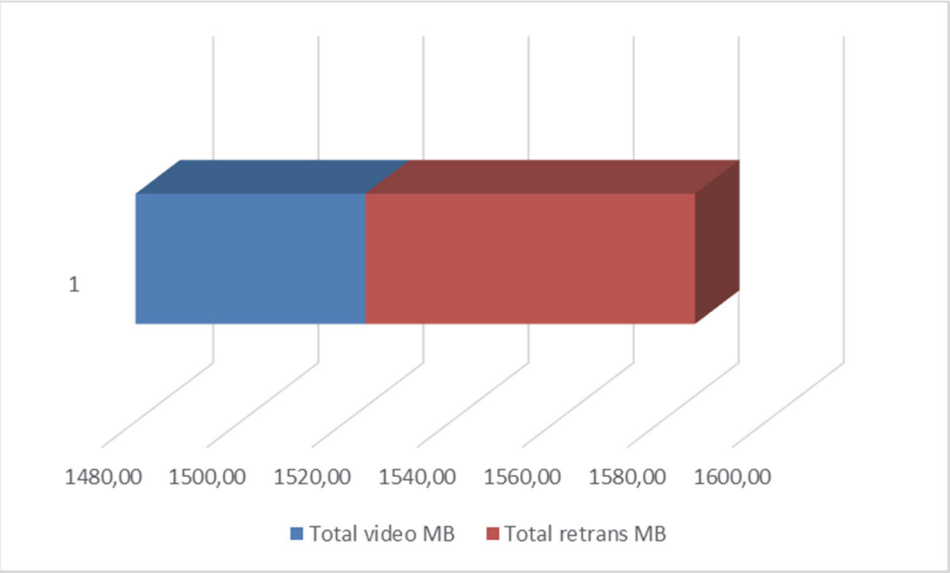

Figura 2.28 Efectos de la retransmisión mejor escenario

En este mejor escenario, donde las interrupciones se han dado casi al inicio del video se puede observar en la Figura 2.27 que el flujo de 14,75 MB ha sido retransmitido 3 veces. En la Figura 2.28 se puede establecer la cantidad de bytes retransmitidos que afectan la prestación del servicio de video streaming en un 4%.

Es importante comparar los efectos de las interrupciones en el peor escenario donde las interrupciones se dan cerca de la finalización del video, para ello se ha tomado datos de una experimentación real (Figuras 2.29 y 2.30), donde el tamaño de los flujos retransmitidos es mucho mayor en comparación a los flujos del mejor escenario y por ende tienen un 50% de afectación a la calidad percibida por el usuario del servicio proporcionado.

En este peor escenario se puede observar en la Figura 2.31 que el flujo de 236,88 MB ha sido retransmitido 3 veces. Este flujo que se retransmite por tres ocasiones es totalmente superior al flujo que se repite igual cantidad en el mejor escenario (14,75 MB). Demostrando la afectación al canal por la cantidad de veces que los flujos se retransmiten, en este caso son inaceptables ya que afectan indiscutiblemente a la QoE y QoS del servicio de video streaming, al ocasionar desperdicio de tiempo por parte del usuario en la reproducción repetida del video.

	Tstart	Tdata1	Tint1	Tstart	Tdata2	Tint2	Tstart	Tdata3	Tint3	Tstart	Tdatan'	Tdatan	a (n+1) * Tstart	b ∑(Tdatai)	c ∑(Tinti)	Texec a+ 2(b) + c + Tdatan'
1	2,43	189,50	10,96	2,43	197,79	8,30	2,43	223,81	13,84	2,43	607,90	1219,00	9,72	611,1	33,1	1872,92
MB	3,04	236,88	13,70	3,04	247,24	10,38	3,04	279,76	17,30	3,04	759,88	1523,75	12,15	763,88	41,38	2341,15

Figura 2.29 Experimentación peor escenario

Tdata	Tamaño bytes	bytes retransmitidos	# retrans	total bytes retrans
223,81	279,76	32,53	1	32,53
197,79	247,24	10,36	2	20,73
189,50	236,88	236,88	3	710,63
Total videos	**Total video MB**	**Total retrans MB**	**%**	
1219	1523,75	763,88	50	

Figura 2.30 Total retransmisión en el peor escenario

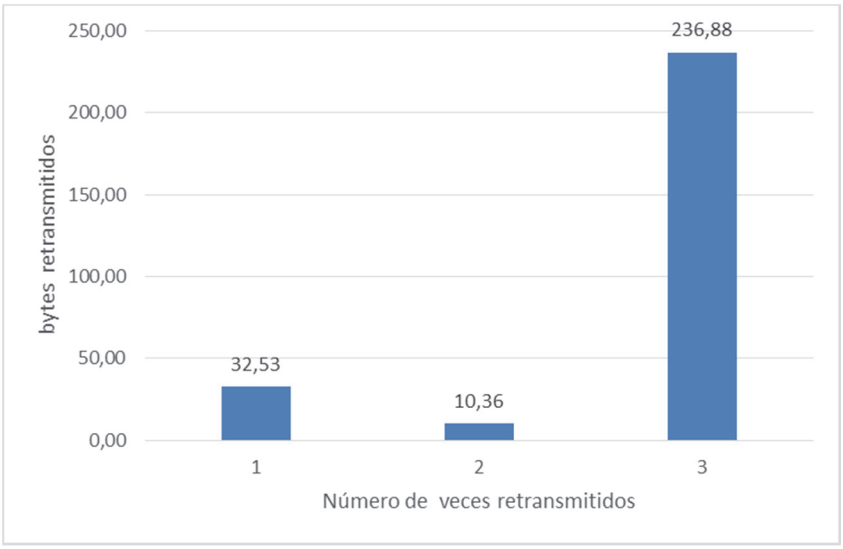

Figura 2.31 Resumen de retransmisiones peor escenario

Se puede analizar el porcentaje adicional del 50% que se paga por causa de las retransmisiones, establecido en la Figura 2.32; debido a que es el peor escenario las interrupciones provocan mayores niveles de retransmisión que afectan a la congestión del canal de transmisión e influyen negativamente en el usuario.

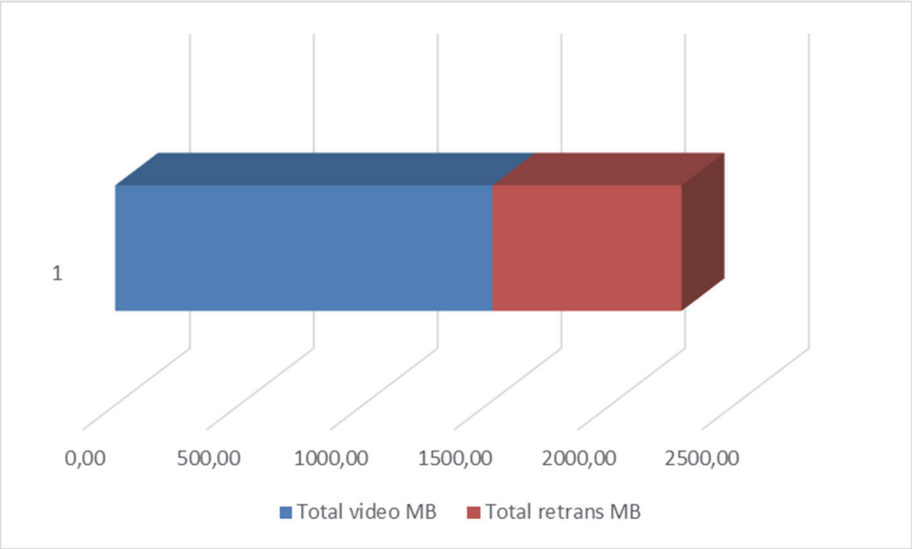

Figura 2.32 Efectos de la retransmisión peor escenario

Para definir mayor sustento en el efecto negativo de las interrupciones, se ha realizado otras pruebas experimentales con un video de duración de 1219 segundos y aplicando 3 interrupciones, las 5 experimentaciones han sido aplicadas para el mejor y peor escenario.

En la figura 2.33 se definen los tiempos de transmisión y los tiempos de interrupción del mejor escenario, donde las interrupciones se han dado cerca del inicio del video, en tanto que en la Figura 2.34 se detalla la experimentación realizada en el peor escenario, en las que las interrupciones se han generado cerca del final del video.

Obteniéndose el análisis de los escenarios en la Figura 2.35, que permite observar claramente la afectación del número de veces que los flujos se retransmiten, principalmente en el peor escenario, provocando mayores tiempos de ejecución del video.

Otro punto que se debe analizar es la duración de una interrupción, este tiempo que no está dispuesto a pagar el usuario y que genera problemas como el abandono de la sesión. Para la elaboración de la Figura 2.36 se han tomado datos reales de una prueba experimental realizada, se deduce que el promedio de duración de una interrupción en este caso es 14 segundos, tiempo elevado que afecta directamente a la satisfacción del usuario.

	Tstart	Tdata1	Tint1	Tstart	Tdata2	Tint2	Tstart	Tdata3	Tint3	Tstart	Tdatan'	Tdatan	a	b	c	Texec
													(n+1) * Tstart	∑(Tdatai)	∑(Tinti)	a+ 2(b) + c + Tdatan'
1	3,93	16,20	12,21	3,93	22,18	10,90	3,93	11,80	16,06	3,93	1168,82	1219,00	15,72	50,18	39,17	1324,07
2	2,83	30,75	8,00	2,83	30,61	9,35	2,83	32,79	9,80	2,83	1124,85	1219,00	11,32	94,15	27,15	1351,62
3	2,66	18,80	11,86	2,66	18,13	11,55	2,66	10,04	14,83	2,66	1172,03	1219,00	10,64	46,97	38,24	1314,85
4	2,80	30,61	21,11	2,80	50,20	9,54	2,80	34,35	8,98	2,80	1103,84	1219,00	11,2	115,16	39,63	1384,99
5	3,06	29,63	12,28	3,06	24,46	13,16	3,06	30,24	10,90	3,06	1134,67	1219,00	12,24	84,33	36,34	1351,91

Figura 2.33 Experimentación mejor escenario

	Tstart	Tdata1	Tint1	Tstart	Tdata2	Tint2	Tstart	Tdata3	Tint3	Tstart	Tdatan'	Tdatan	a	b	c	Texec
													(n+1) * Tstart	∑(Tdatai)	∑(Tinti)	a+ 2(b) + c + Tdatan'
1	2,43	189,50	10,96	2,43	197,79	8,30	2,43	223,81	13,84	2,43	607,90	1219,00	9,72	611,1	33,1	1872,92
2	2,70	182,76	12,69	2,70	152,16	8,81	2,70	234,81	15,38	2,70	649,27	1219,00	10,8	569,73	36,88	1836,41
3	4,70	73,83	14,99	4,70	130,88	12,23	4,70	83,13	18,54	4,70	931,16	1219,00	18,8	287,84	45,76	1571,40
4	2,28	144,01	12,29	2,28	147,45	11,63	2,28	217,51	11,78	2,28	710,03	1219,00	9,12	508,97	35,7	1772,79
5	5,05	114,44	22,90	5,05	123,48	12,14	5,05	157,61	16,38	5,05	823,47	1219,00	20,2	395,53	51,42	1686,15

Figura 2.34 Experimentación peor escenario

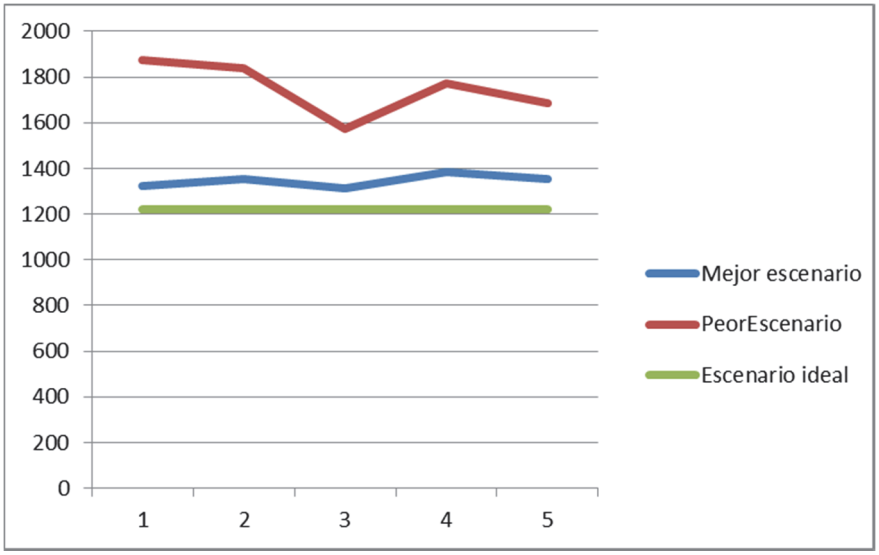

Figura 2.35 Análisis tiempo de ejecución

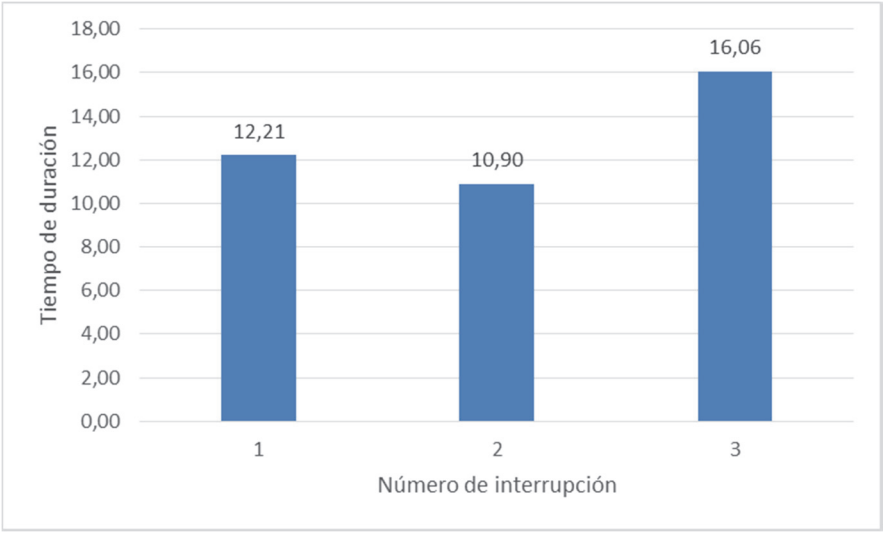

Figura 2.36 Análisis de interrupciones

Para realizar las pruebas experimentales y verificar el modelo de tiempo diseñado, procedimos a realizar 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, por lo que era necesario volver a abrir una sesión nueva. Por todo lo expuesto, decidimos utilizar una aplicación propia que si nos permitía medir el efecto de estas reconexiones, que es la transmisión en tiempo real basado en sensores.

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 inicio de sesión (*Tstart*), en los que se transmiten datos (*Tdatai*) y en los que no hay tal transmisión (*TInti*). Basado en ello se han realizado 3 pruebas aplicando 3 interrupciones, en la Figura 2.37 puede analizarse los datos obtenidos de forma real y en la Figura 2.38 los datos calculados utilizando el modelo de tiempo, estableciendo la efectividad de cálculo de tiempo de ejecución, al obtenerse valores con poca diferencia a los reales.

Se puede observar en la Figura 2.39, la afectación clara de las interrupciones en el tiempo de ejecución del video streaming, generando hasta el 100% adicional del tiempo de duración del video lo que afecta a la QoE del cliente.

Prueba	tiempo 1	tiempo 2	tiempo 3	tiempo 4	tiempo 5	tiempo 6	tiempo 7	tiempo 8	Duración Total	Duración segundos
1	00:04,19	00:27,95	00:44,29	00:38,98	00:33,01	00:38,54	00:43,05	00:10,64	04:00:69	240,69
2	00:04,25	00:33,54	00:32,58	00:42,98	00:38,86	00:31,00	00:23,08	00:10,09	03:36,41	216,41
3	00:03,71	00:34,06	00:20,06	00:31,81	00:18,23	00:34,00	00:32,09	00:10,50	03:04,50	184,50

Figura 2.37 Tiempos experimentación real

Prueba	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b	c	Texec	Diferencia
									∑(Tdatai)	∑(Tinti)	Tstart + b + c + Tdatan	
1	4,19	27,95	44,29	38,98	33,01	38,54	43,05	10,64	105,47	120,35	240,65	0,04
2	4,25	33,54	32,58	42,98	38,86	31,00	23,08	10,09	107,52	94,52	216,38	0,03
3	3,71	34,06	20,06	31,81	18,23	34,00	32,09	10,50	99,87	70,38	184,46	0,04

Figura 2.38 Tiempos aplicación modelo de tiempo

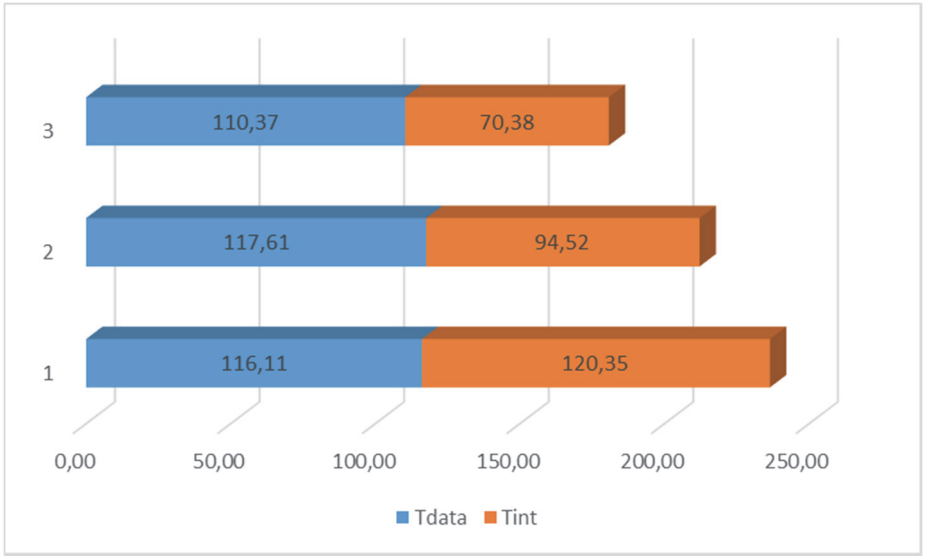

Figura 2.39 Relación entre tiempos de transmisión y de interrupción

3. Solución básica basada en proxies para iOS

Se expone a profundidad el diseño de la solución básica, su caracterización basada en patrones software para implantar un mecanismo proactivo que controla interrupciones en la transmisión de video streaming. De manera puntual, se detalla el proceso de desarrollo del prototipo desarrollado denominado JADE-iPhone, que implementa el modelo básico propuesto y se demuestra su efectividad al brindar continuidad de la transmisión del video en el dispositivo móvil iOS.

3.1. Análisis del diseño basado en patrones software

En el capítulo anterior se explicó el mecanismo básico que permite la ejecución del proceso de reproducción de video a través de uso del patrón MVC, en base al análisis y pruebas realizadas del modelo matemático se definió el impacto de las interrupciones en la cantidad de retransmisiones de los datos que pueden ocurrir, influyendo en la congestión de la red y en el tiempo de ejecución del servicio video streaming. El impacto negativo de mayor preocupación, es la QoE del usuario al causarle molestias de demoras, mala calidad de visualización, reinicio de sesión que provoca el abandono de la sesión. Estamos interesados en establecer una solución basada en proxies que controla las interrupciones y reduce las retransmisiones de los flujos, esto se logra al ser proactivos en detectar una eminente quiebre en la comunicación y establecer los mecanismos necesarios para permitir al usuario continuar visualizando el video desde la posición de reproducción en la que se encontraba cuando ocurrió la interrupción. Buscamos demostrar que el modelo propuesto reduce las retransmisiones y mejora el tiempo de ejecución, lo que apoya indudablemente a la mejora de la QoE del usuario, también pretendemos conseguir medir el impacto de mejora de transmisión del canal lo que influye en la QoS.

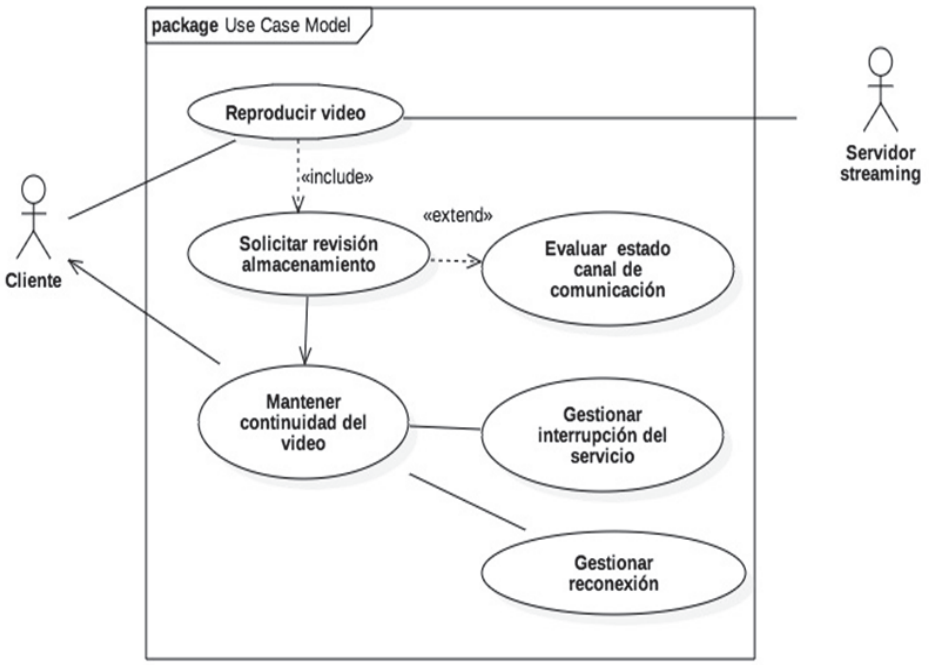

Figura 3.1 Diagrama de casos de uso

Se han establecido nuevas funcionalidades a este modelo con la finalidad de controlar las interrupciones que puedan darse en el canal de comunicación y lograr la reconexión automática, potenciando el servicio de video streaming al proveer de mejores niveles de QoE al usuario al proveer continuidad de reproducción al ocurrir problemas en el canal de comunicación.

El modelo propuesto mantiene la flexibilidad al cambio, orden, reutilización, escalabilidad, modularidad… aspectos solventados por uso del patrón MVC en el mecanismo básico de video streaming, descrito en el capítulo anterior. Los requisitos que debe cubrir la propuesta se han establecido a través del modelo de casos de uso definido en la Figura 3.1 en el que se establecen los procesos de la reproducción del video, revisión del almacenamiento, análisis del canal de comunicación, gestión de una interrupción y manejo de una reconexión.

En la tabla 3.1 se detalla todo el proceso de la solicitud del servicio de video streaming, la coordinación por parte del *Proxy* y la atención del pedido realizada por el servidor.

Tabla 3.1 Caso de uso reproducir video

Caso de Uso: **reproducir video**
Actores: cliente
Resumen: se describe puntualmente como inicia la petición de la reproducción del video y su despliegue.
Precondiciones:
Descripción: Curso Normal: 1. El cliente solicita la visualización del video 2. Ingresa los datos del puerto y servidor 3. Los datos son validados por el *Proxy* 4. Al no existir problemas el *Proxy* solicita el servicio al servidor de video streaming 5. Se despliega una ventana en la que inicia el video streaming 6. El servidor inicia la entrega periódica de los flujos al almacenamiento temporal
Pos condiciones: se solicita la revisión continua del almacenamiento
Observaciones:

Se detalla la funcionalidad referente a la revisión periódica del almacenamiento temporal donde se alojan los frames para determinar si hay problemas cuando se supera el umbral permitido (tabla 3.2) y todo el proceso relacionado con evaluar el estado del canal de comunicación inalámbrico y establecer si se ha dado una interrupción para alertar de este evento al *Proxy* quien realiza las acciones tendientes a enfrentar este problema (tabla 3.3).

Tabla 3.2 Caso de uso solicitar revisión almacenamiento

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

Tabla 3.3 Caso de uso evaluar estado canal de comunicación

Caso de Uso: **evaluar estado canal de comunicación**
Actores:
Resumen: para asegurar que se trata de una interrupción es necesario evaluar cómo está la comunicación inalámbrica.
Precondiciones: el *Proxy* solicita la evaluación del canal de comunicación
Descripción: Curso Normal: 1. Se evalúa la comunicación a nivel de RTSP 2. Se evalúa la comunicación a nivel de UDP 3. Se determina el estado del canal de comunicación 4. Este estado es comunicado al *Proxy*
Pos condiciones: se solicita mantener la continuidad del servicio
Observaciones:

Se explica el proceso que sigue el *Proxy* para definir la operación a realizar y solventar una interrupción dada (tabla 3.4) y se detalla toda la funcionalidad al ocurrir una interrupción, el uso de buffer para almacenar los frames que han sido enviados por el servidor y que no pueden ser entregados al cliente al haber ocurrido una interrupción (tabla 3.5).

Tabla 3.4 Caso de uso mantener continuidad del video

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

Tabla 3.5 Caso de uso gestionar interrupción del servicio

Caso de Uso: **gestionar interrupción del servicio**
Actores: cliente
Resumen: explica el detalle de las actividades a cumplir cuando ocurre una interrupción y como se controla la misma
Precondiciones:
Descripción: Curso normal : 1. Se solicita el almacenamiento de los flujos en el buffer 2. Se despliega en la interfaz gráfica del usuario el mensaje que se ha dado un problema 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 Caso de uso gestionar reconexión

Caso de Uso: **gestionar reconexión**
Actores:
Resumen: explica el detalle de las actividades a cumplir cuando el canal se ha restablecido
Precondiciones:
Descripción: Curso normal : 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:

En la tabla 3.6 se explica la recuperación de los frames del buffer y el despliegue en el cliente una vez que la comunicación se ha restablecido.

3.1.1. Justificación de los patrones software a utilizar

La propuesta de este diseño se fundamenta en proveer un modelo genérico que pueda ser implementado en un escenario de experimentación específico para dispositivos inalámbricos, se ha decidido utilizar el patrón MVC aplicado en el modelo del capítulo anterior, por las potencialidades que presenta al permitir desarrollar código ordenado al separar los datos de la aplicación, la interfaz de usuario y la lógica de control, lo que posibilita la facilidad de cambios y adición de funcionalidades, apoya a la integración y la reutilización de componentes, brinda escalabilidad y facilidad del desarrollo de prototipos. Se ha decidido implementar el patrón *Composite* en el lado de la vista para que su contexto y componentes se organicen estructuradamente y puedan ser reemplazados con facilidad; el patrón *Observer* acompaña al modelo para establecer cambios de estado del almacenamiento o buffer y notificar oportunamente al controlador de los cambios de estado del objeto observado y se utiliza el patrón *Strategy* en la capa controladora para determinar las acciones a seguir en caso de gestionar una interrupción o reconexión del servicio.

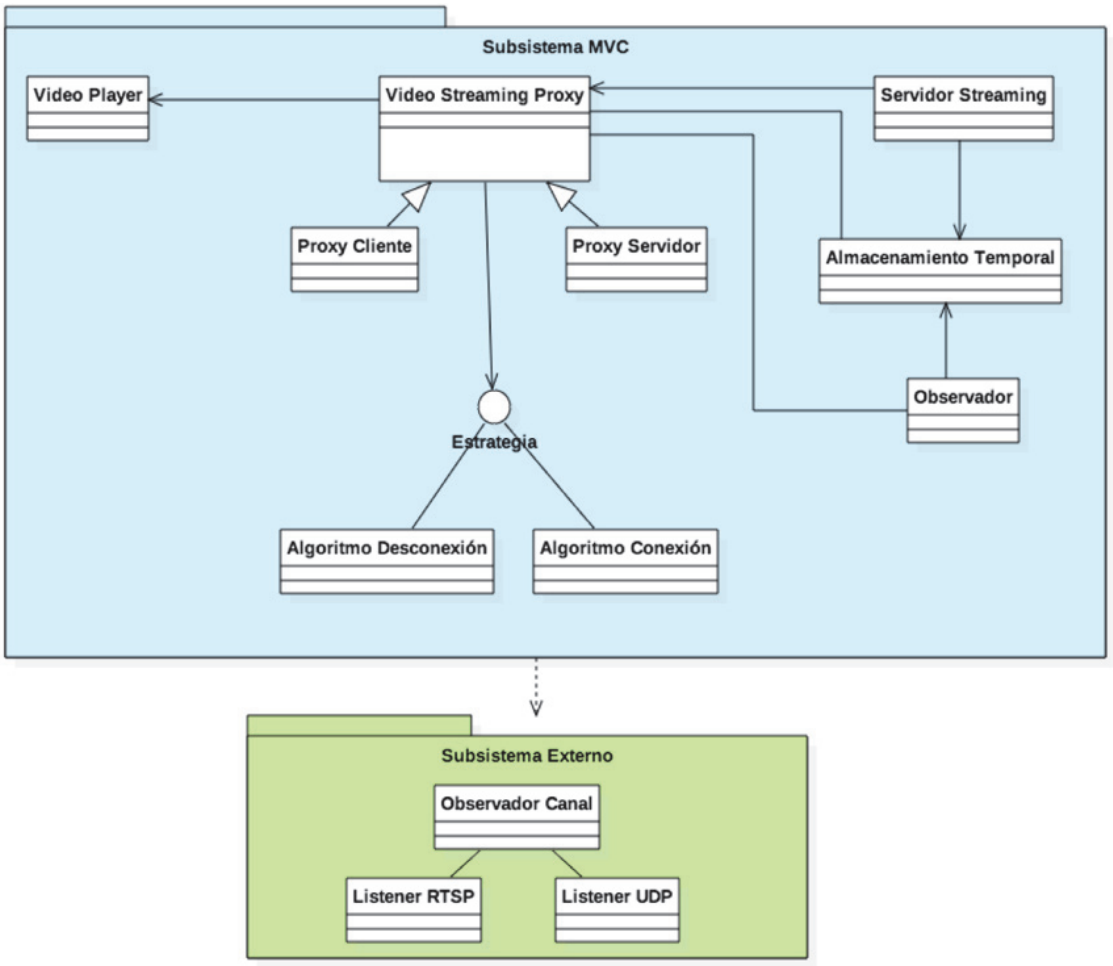

Figura 3.2 Modelo basado en patrones

El patrón *Proxy* es el encargado de ser el intermediario entre el cliente y el servidor se ubica en la capa controladora, quien es el orquestador de todo el mecanismo de control de interrupción. Para lograr la funcionalidad requerida y permitir el control de las interrupciones se han adicionado componentes que pueden observarse en la Figura 3.2, en este se puede observar el incremento de un subsistema externo que permite se ejecute la evaluación del canal de comunicación.

Los requisitos establecidos en el caso de uso *solicitar revisión almacenamiento* se ha implementado a través del patrón *Observer*, que fortalece la capa del modelo del MVC, se encarga de estar constantemente monitoreando al almacenamiento y realiza la evaluación del estado del buffer vs el nivel máximo establecido. Cuando el buffer supera el umbral el observador notifica al *Proxy* este evento.

Cuando ocurre la notificación de problemas en el almacenamiento, se debe ejecutar los requisitos definidos en el caso de uso *evaluar estado canal de comunicación* para ello se ha visto la necesidad de crear un paquete externo conformado por un observador al mismo nivel que el controlador, este se encarga de recibir el pedido del *Proxy* y revisar que sucede con el canal inalámbrico, la potencialidad de este observador está en que garantiza al aplicativo si realmente hay una interrupción, hecho que es notificado al servidor. De igual forma cuando hay una reconexión, la utilidad de este observador es visible al proporcionar el servicio de indicar de forma oportuna y eficiente cuando se restablece el canal.

Para cumplir con los requisitos especificados en los casos de uso *mantener continuidad del servicio*, *gestionar interrupción del servicio* y *gestionar reconexión* al obtener como respuesta el estado de canal desconectado, el *Proxy* solicita al patrón *Strategy*, la ejecución del algoritmo de interrupción, el cual realiza dos actividades: solicitud del almacenamiento temporal de los flujos emitidos por el servidor y despliegue en la vista del mensaje *reconectando*.

Cuando los problemas de conexión han sido superados, el observador del subsistema externo determina que el canal de comunicación se ha restablecido comunicando de este hecho al *Proxy*, quién solicita al patrón *Strategy* la ejecución del algoritmo de conexión; en este caso se pide que los flujos almacenados en el almacenamiento temporal sean transmitidos al cliente.

3.1.2. Diseño arquitectónico y de despliegue

En el diseño se define la arquitectura de la propuesta que soporte los requisitos y nos lleve a implantar el patrón básico basado en patrones software. El diagrama de clases del sistema se expone en la Figura 3.3.

A continuación se explica la relación entre el modelo de patrones software y las clases que 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: *ServerStreaming*, *Buffer* y *DataListener*.
- *Vista y patrón Composite*: las clases que son ejecutadas para lograr el despliegue del video streaming en el dispositivo cliente son: *VideoPlayer*, *URLStreaming* y *VideoView*.
- *Controlador*: define el patrón *Proxy* el cual está representado por las clases *AgentProxy*, *ServerAgentProxy* y *ClienteAgentProxy*. El patrón *Strategy* también fortalece la capa de control implementado mediante las clases: *ActionControler*, *InterrupciónAction* y *ReconexionAction*.
- El observador externo que permite evaluar el estado del canal de comunicación está representado por las clases: *EnlaceListener*, *ListenerUDP* y *ListenerRTSP*.

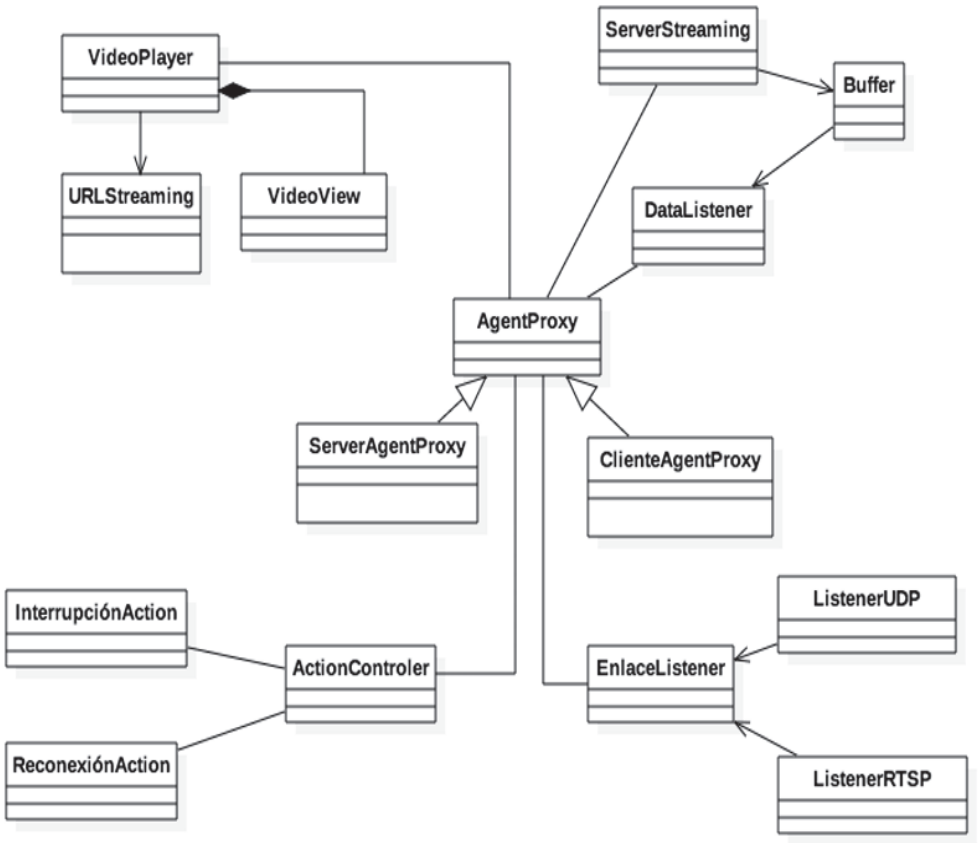

Figura 3.3 Diagrama de clases

En la Figura 3.4 se muestra el diagrama de secuencia. En este diagrama se observa la comunicación que se da entre las clases establecidas anteriormente para solventar la solicitud y despliegue del servicio de video streaming, una vez establecido este enlace es necesario activar la escucha del control de datos. Si el almacenamiento presenta nivel superior al umbral, este evento es notificado a la capa controladora.

Para manejar una interrupción, es importante garantizar que se ha dado efectivamente la misma, por lo que al darse una alerta de almacenamiento, se activa el observador del estado del canal de comunicación. Al definir la existencia de una interrupción, se activan las solicitudes necesarias para que los flujos sean almacenados en el buffer temporal.

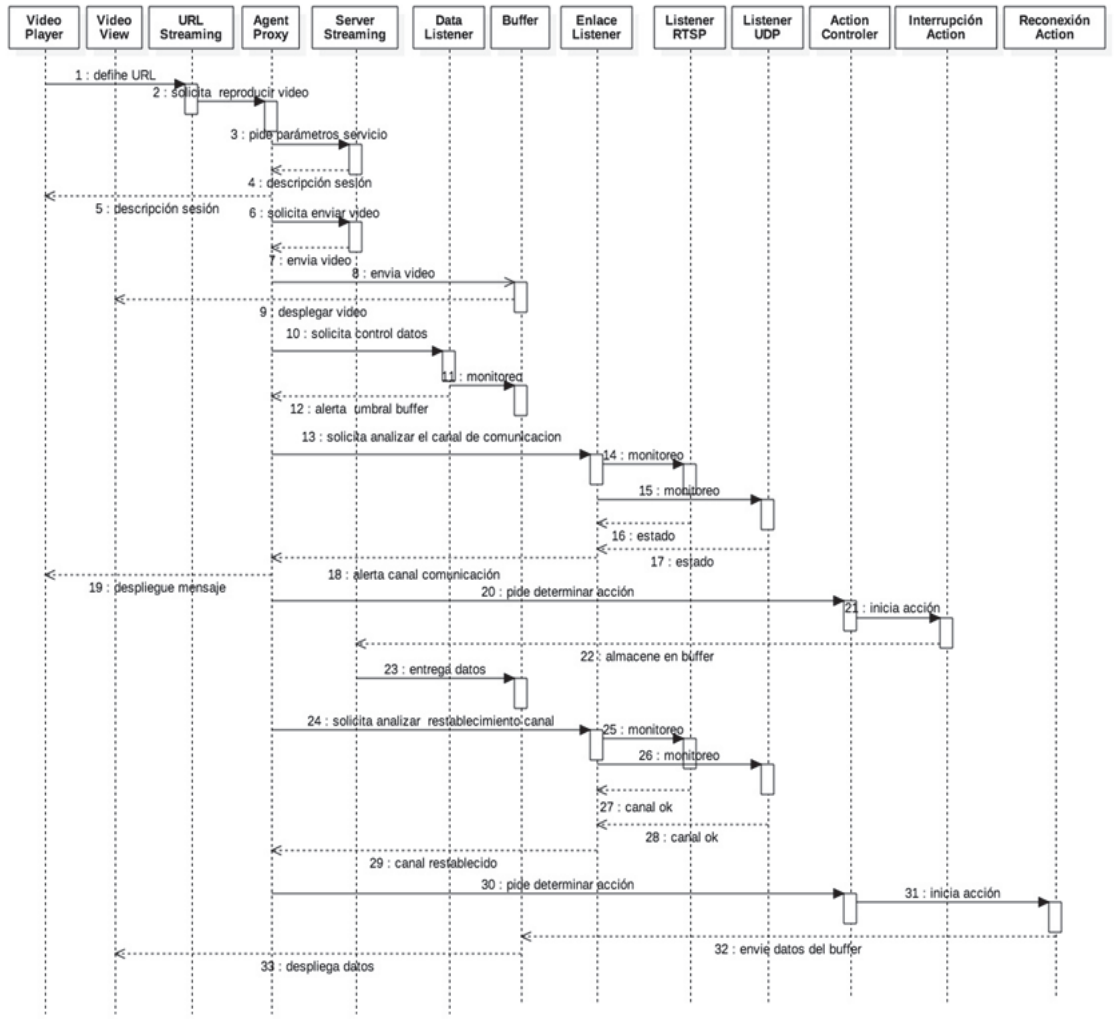

Figura 3.4 Diagrama de secuencia

Una vez que el observador indique que el canal se ha restablecido, otras acciones son ejecutadas para visualizar los flujos almacenados en el dispositivo cliente.

En la Figura 3.5 se puede observar el diagrama de despliegue, el dispositivo móvil aloja al *VideoPlayer*, *URLStreaming* y *VideoView* que son las clases que inician la solicitud del pedido del servicio de video streaming, esta pedido es recibido por el *Agente ProxyCliente* y enviado al *AgenteProxyServidor* a través de comunicación ACL, esta solicitud es direccionada al *ServerStreaming*, el cual devuelve el flujo hacia el *Proxy*, luego los flujos pasan a ser almacenados en el *Buffer* del dispositivo móvil e inicia su despliegue en el cliente.

Al iniciar esta visualización el *Proxy* solicita el trabajo de observador a *DataListener* quien continuamente está revisando al *Buffer* y define alerta de umbral superado. Si este evento ocurre el *Proxy* solicita el servicio de *EnlaceListener* de observar el estado del canal de comunicación, 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*.

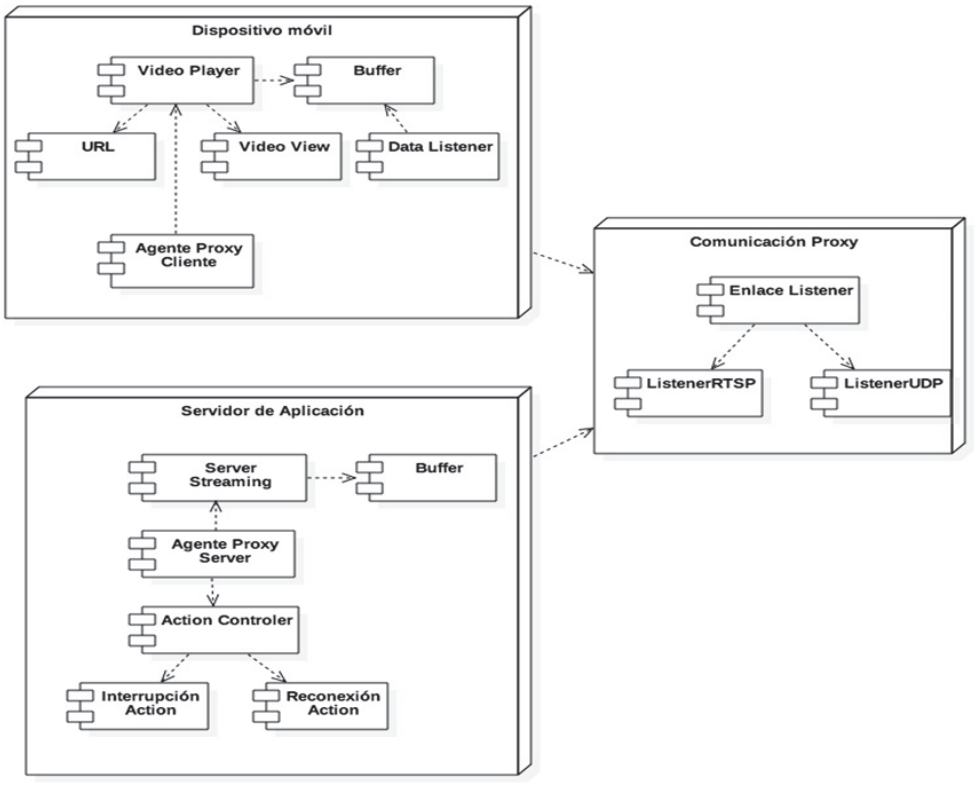

Figura 3.5 Diagrama de despliegue

Este algoritmo se encarga de que el *Proxy* solicite al *ServerStreaming* el almacenamiento temporal de los flujos que no pueden ser transmitidos en el *Buffer* y al *EnlaceListener* pide que evalúe el canal y determine su restablecimiento. Al restablecerse el canal, el *Proxy* recibe esta notificación y pide al *ActionControler* la ejecución del algoritmo correspondiente, para esta situación sería *ReconexiónAction* 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.

3.2. Modelo matemático de rendimiento

A continuación se explica el detalle de la definición de la ecuación del tiempo total de ejecución, la cual se basa en la ecuación obtenida en el mecanismo básico de video streaming analizado en el capítulo anterior.

$$T_{exec} = (n+1) \times T_{start} \quad + 2\sum_{i=1}^{n}\left(T_{data_i}\right) + \sum_{i=1}^{n} T_{int_i} + T'_{data_n}$$

Donde:

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

$\sum_{i=1}^{n}\left(T_{data_i}\right)$ es el tiempo total de *Tdata* transmitidos más de una vez.

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

T'_{data_n} es el tiempo del dato transmitido una sola vez.

Esta solución básica basada en proxies, ha establecido la funcionalidad de dos observadores: el primero que realiza el monitoreo constante del buffer ubicado en el dispositivo móvil para determinar si el umbral es superior al permitido; el segundo observador se encarga de evaluar el canal de comunicación y definir proactivamente una interrupción. Esta interrupción es controlada mediante la transmisión de los frames

del video hacia un almacenamiento temporal hasta que puedan ser reenviados una vez la comunicación se restablezca.

Con esta nueva funcionalidad, el cliente no tiene que solicitar la visualización del video y esperar a que se despliegue el primer frame cada que ocurra una interrupción, esto determina que durante todo el tiempo de ejecución se realizara un único *tstart*, que se da al inicio de la sesión de video streaming, por lo que se elimina todos los *tstart* adicionales por cada interrupción ocurrida, la ecuación al eliminar *(n+1)* es la siguiente.

$$T_{exec} = T_{start} \quad + 2\sum_{i=1}^{n}\left(T_{data_i}\right) + \sum_{i=1}^{n} T_{int_i} + T'_{data_n} \quad (3)$$

Los nuevos patrones software establecidos en este modelo apoyan en el control eficiente de la interrupción, identificándola proactivamente y gestionando la grabación de los frames que no pueden ser transmitidos a un almacenamiento temporal, una vez que se restablece la comunicación, los frames almacenados son reproducidos en el dispositivo cliente, proporcionando al usuario continuidad de la visualización del video desde la posición en la que se encontraba al ocurrir la interrupción; lo que confluye a la no existencia de retransmisiones.

Si el total de retransmisiones se representa por la ecuación:

$$T_{retransmision} = \sum_{i=1}^{n}\left(T_{data_i}\right)$$

Al eliminar el total de retransmisiones de la ecuación 3 obtenemos la ecuación.

$$T_{exec} = T_{start} \quad + \sum_{i=1}^{n}\left(T_{data_i}\right) + \sum_{i=1}^{n} T_{int_i} + T'_{data_n}$$

Al no existir retransmisiones no se requiere contabilizar T'_{data_n} que es el tiempo del dato transmitido una sola vez, más bien debe ser reemplazado por el último frame determinado como T_{data_n} al modificar la ecuación obtenemos finalmente ecuación del tiempo de ejecución del modelo propuesto.

$$T_{exec} = T_{start} \ + \sum_{i=1}^{n} \left(T_{data_i}\right) + \sum_{i=1}^{n} T_{int_i} \ + T_{data_n}$$

Donde:

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

$\sum_{i=1}^{n} \left(T_{data_i}\right)$ es el tiempo total de Tdata transmitidos.

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

T_{data_n} es el último frame transmitido.

Con un video de duración de 823 segundos y ejecutado 3 interrupciones, se han realizado pruebas en dos tipos de escenarios: mejor y peor. En el mejor escenario se han generado las interrupciones cerca del inicio del video, Figura 3.6. En el peor escenario las interrupciones se han dado cerca del final del video, Figura 3.7.

Como puede observarse en la Figura 3.8, no se dan grandes diferencias entre los dos escenarios, la razón de ello es que al tiempo de ejecución influye la sumatoria de las interrupciones, las cuales está en el rango de 12 a 18 segundos, concluyendo que no influye en qué posición del video se genere la interrupción sino su duración.

	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b ∑(Tdatai)	c ∑(Tinti)	Texec Tstart + b + c + Tdatan
1	3,93	16,20	12,21	22,18	10,90	11,80	16,06	772,82	50,18	39,17	866,10
2	2,80	30,61	21,11	50,20	9,54	34,35	8,98	707,84	115,16	39,63	865,43
3	2,66	18,80	11,86	18,13	11,55	10,04	14,83	776,03	46,97	38,24	863,90
4	3,06	29,63	12,28	24,46	13,16	30,24	10,90	738,67	84,33	36,34	862,40
5	2,83	30,75	8,00	30,61	9,35	32,79	9,80	728,85	94,15	27,15	852,98

Figura 3.6 Experimentación mejor escenario

	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b ∑(Tdatai)	c ∑(Tinti)	Texec Tstart + b + c + Tdatan
1	2,28	144,01	12,29	147,45	11,63	217,51	11,78	314,03	508,97	35,70	860,98
2	2,43	189,50	10,96	197,79	8,30	223,81	13,84	211,90	611,10	33,10	858,53
3	4,70	73,83	14,99	130,88	12,23	83,13	18,54	535,16	287,84	45,76	873,46
4	5,05	114,44	22,90	123,48	12,14	157,61	16,38	427,47	395,53	51,42	879,47
5	2,70	182,76	12,69	152,16	8,81	234,81	15,38	253,27	569,73	36,88	862,58

Figura 3.7 Experimentación peor escenario

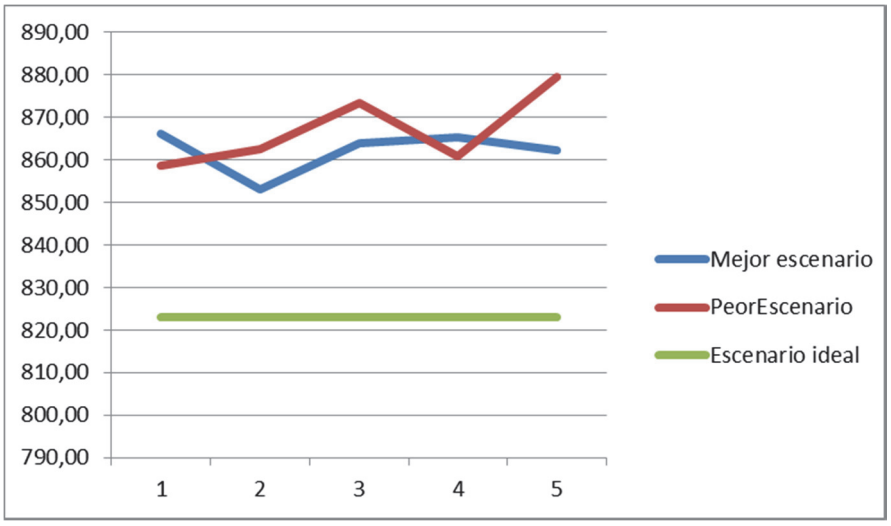

Figura 3.8 Análisis tiempo de ejecución

Con lo expuesto se demuestra el ahorro de tiempo de ejecución en un video administrado por proxies para controlar una interrupción debido a que no ocurren retransmisiones.

Para la ejecución de este modelo ha sido necesario adicionar un almacenamiento temporal donde se alojaran los frames mientras dure una interrupción. En la transmisión del video donde han ocurrido n interrupciones es tomada aquella de mayor duración, en este caso $Tint_1$ = 15.

Para evaluar la cantidad de memoria utilizada, se define como velocidad del canal de 10Mbps en ADSL entonces se determina el número de bits que transmite con la siguiente ecuación:

$$Número\ de\ bits = velocidad\ del\ canal * tiempo\ de\ la\ interrupción$$

$$Número\ de\ bits = 15 * 10 = 150\ \ bits$$

Para obtener bytes lo dividimos para 8, obteniendo 150/8 = 18,75 MB.

En la actualidad existen dispositivos inalámbricos con una memoria de 120MB o superior, el uso de 18,75 MB de almacenamiento es un costo razonable a pagar al recibir los beneficios proporcionados por un mecanismo que controla interrupciones. Además, tal cantidad de memoria no afecta a los grandes almacenamientos locales o en la nube que existen en la actualidad.

3.3. Análisis de la implantación experimental

Se ha desarrollado JADE-iPhone para la plataforma iOS, para implantar el mecanismo que controla la transmisión del video y su visualización en los dispositivos inalámbricos, la idea básica es evitar que el usuario tenga que reiniciar una sesión RTSP, cuando ocurre una interrupción y que pueda recibir en diferido los frames de video perdidos, lo cuales están temporalmente almacenados en un buffer hasta que el dispositivo móvil vuelva al área de cobertura. En el desarrollo de JADE para iPhone se utiliza una arquitectura de agentes de software, que proveen métodos orientados a establecer la reanudación automática de la sesión del usuario sin pérdida de la información.

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

En el diseño se define la arquitectura de la propuesta, los diagramas de clases del sistema se exponen en las Figura 3.9. A continuación se explica la relación entre el modelo de patrones software y las clases que 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: *MemoryStream*, *DataStream* y *NSStream*.
- Vista y patrón *Composite*: las clases que son ejecutadas para lograr el despliegue del video streaming en el dispositivo cliente *EAGLView*, *Render* y *UrlRTSP*.
- Controlador: en este se define el patrón *Proxy* el cual está representado por las clases *Agent*, *AgentProxy* y *NAgent*. El patrón *Strategy* también fortalece la capa de control implementado mediante las clases: *BehaviourFactory*, *BehaviourConnect* y *BehaviourDisconnect*.
- El observador externo que permite evaluar el estado del canal de comunicación está representado por las clases: *ConnectionListener*, *ListenerPortUDP*, *ListenerPortRTSP*, *ListenerMessageUDP*, *ACLMessage*, *ListenerMessageRTSP* y *MessageRTSP*.

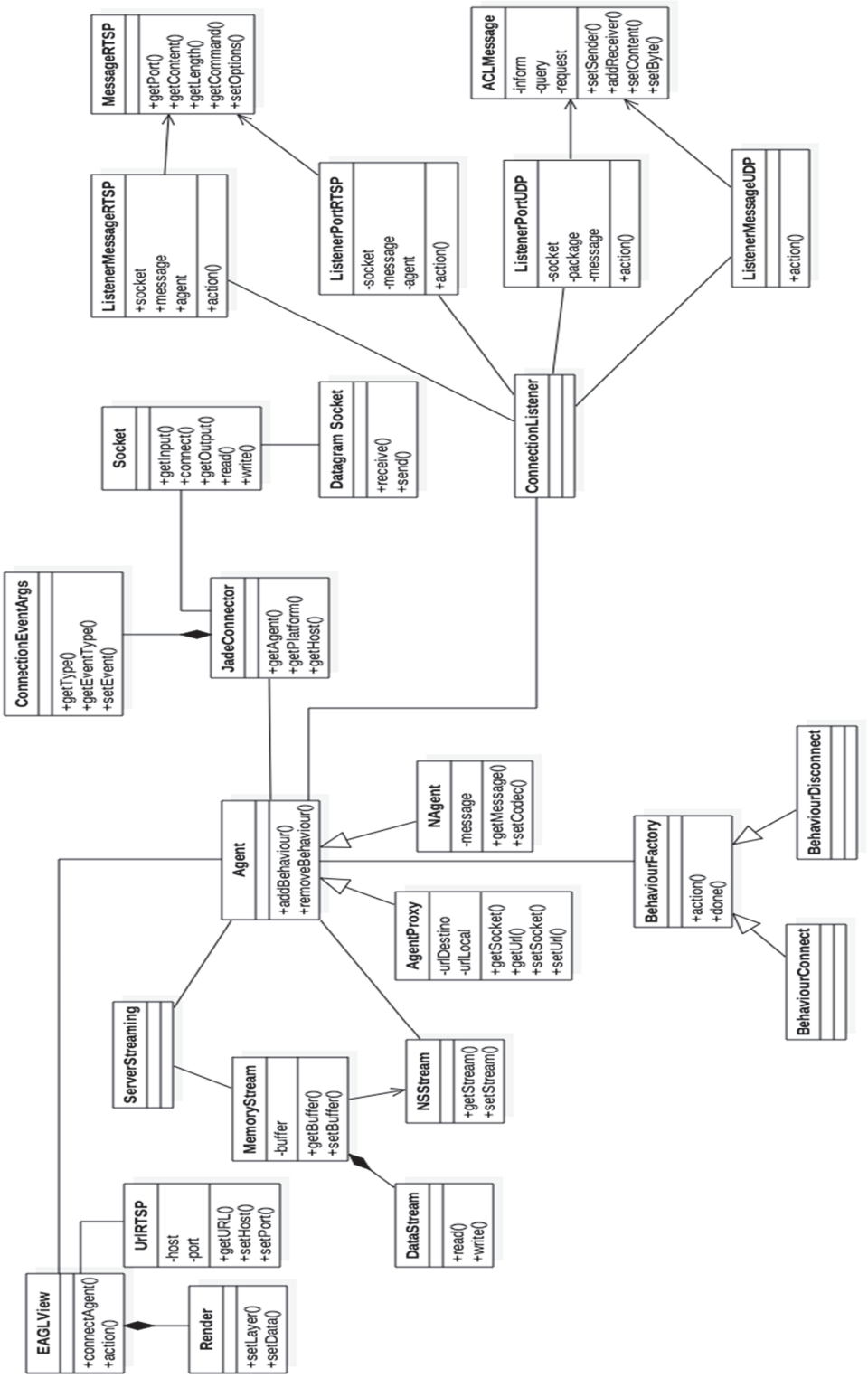

Figura 3.9 Diagrama de clases iPhone

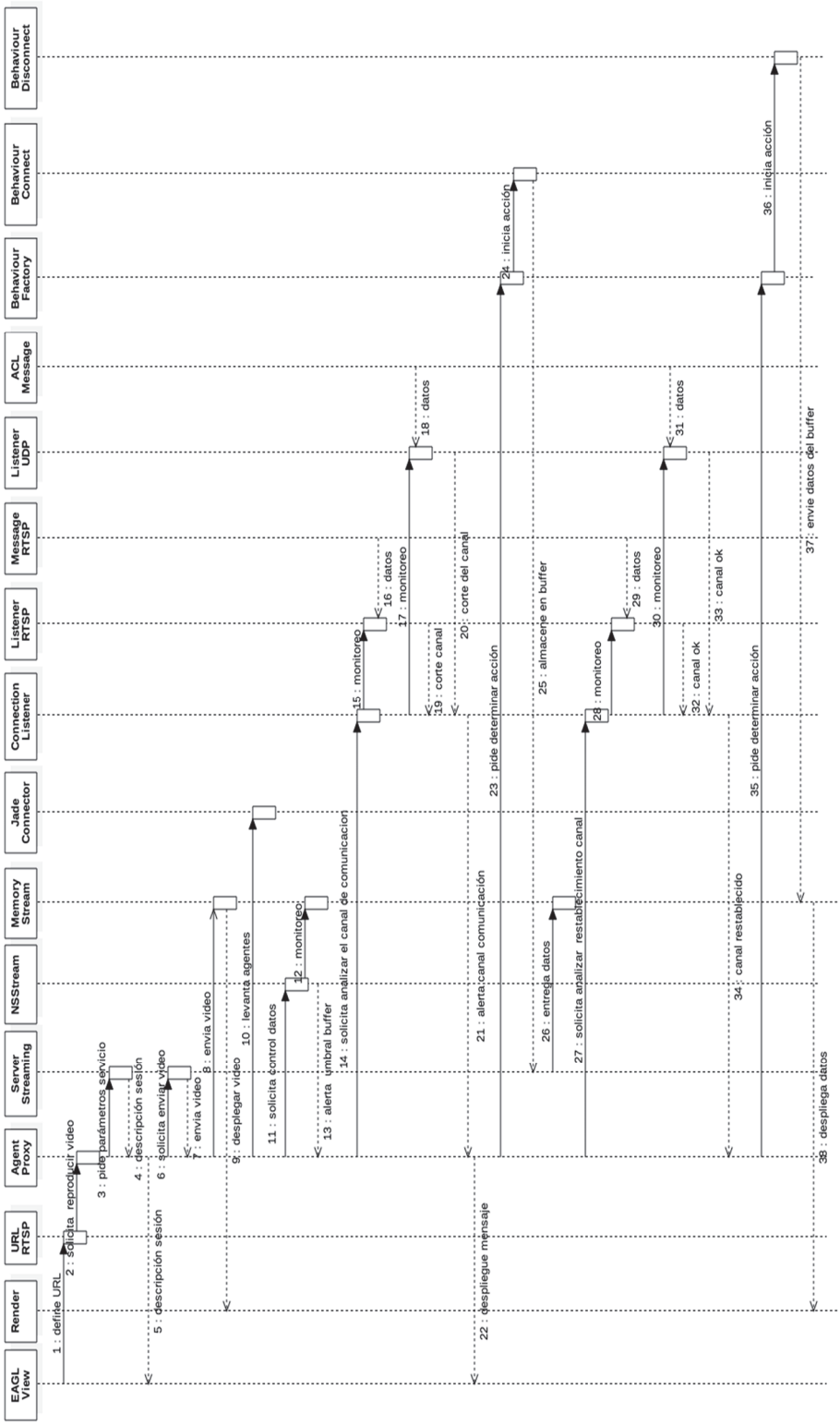

Figura 3.10 Diagrama de secuencia

En la Figura 3.10 se muestran el diagrama de secuencia que 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 ejecuta de acuerdo al evento ocurrido.

En la Figura 3.11 se expone el diagrama de despliegue donde se indica la ubicación de los componentes y la comunicación entre ellos, se puede notar que el *Proxy* se ubica tanto en el cliente como en el servidor de aplicación para conseguir generar la comunicación y gestión del *Buffer* cuando ocurra la interrupción, esto apoyara a que no se pierdan los frames y que sean mostrados en el cliente una vez solucionado el problema que ocasiono la interrupción.

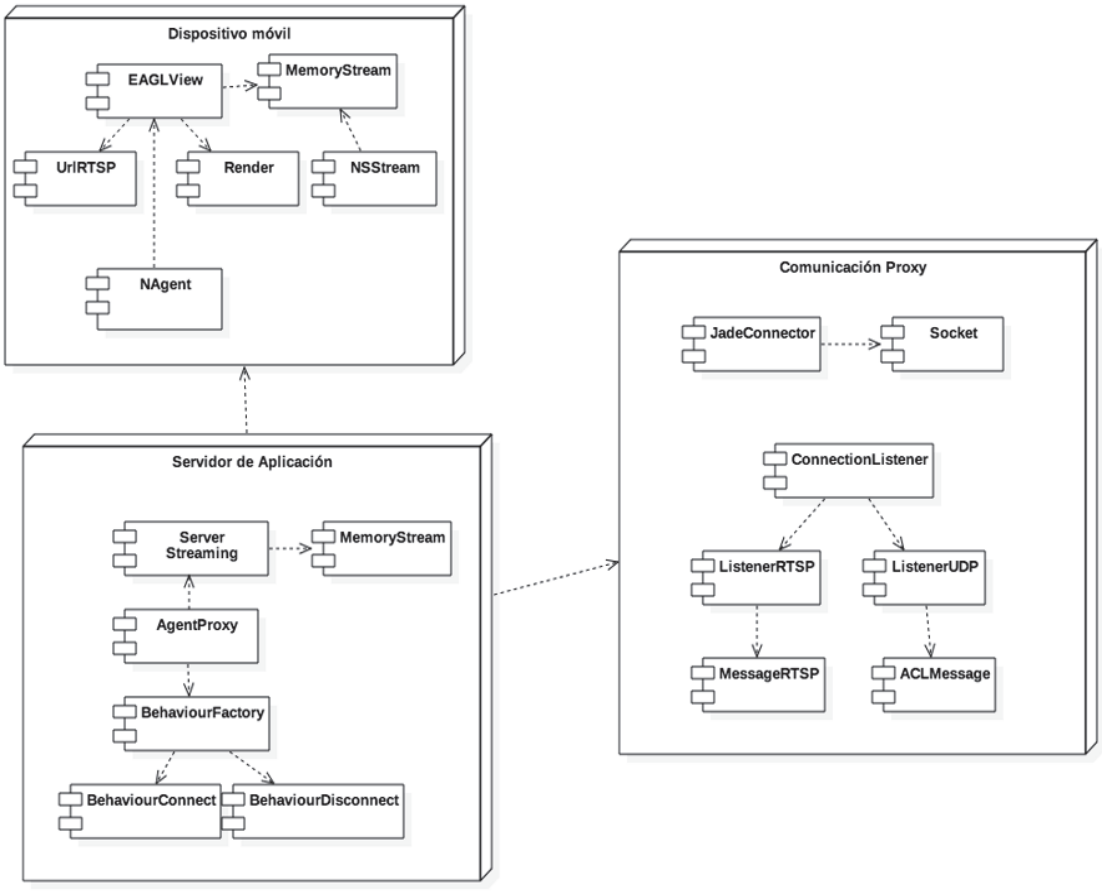

Figura 3.11 Diagrama de despliegue

Al iniciar esta visualización el *Proxy* solicita el trabajo de observador a *NSStream* quien continuamente está revisando a *MemoryStream* 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 *BehaviourFactory* para que seleccione el algoritmo a ejecutar, en este caso *BehaviourDisconnect*. 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 *BehaviourFactory* la ejecución del algoritmo correspondiente, para esta situación sería *BehaviourConnect* 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 propietario

Con miras a crear y ejecutar agentes en el entorno iOS, se desarrolló el aplicativo denominado JADE-iPhone, que es un port del código de JADE SHARP creado para ser utilizado con.NET CF o.NET JADE_SHARP. Este código está bajo licencia *GNU Library General Public License* (*LGPL*) 3 y es compatible con la versión 2.1 de JADE SHARP, la librería desarrollada toma en cuenta los copyright y licencias de JADE.

JADE-iPhone es un aplicativo desarrollado en código abierto para interactuar específicamente con el iOS y al igual que JADE, posibilita la implementación de agentes bajo los estándares FIPA, su primera versión permite la comunicación de mensajes ACL; funciona como JADE LEAP en modo Split, significa que el contenedor en el que se despliega el agente se divide en una parte frontal (FrontEnd) en el dispositivo móvil y una parte trasera (BackEnd) en un servidor remoto.

JADE-iPhone permite el desarrollo y ejecución de agentes JADE en terminales móviles conectados a través de redes inalámbricas, tomando en cuenta que estos dispositivos presentan limitaciones en algunos recursos como es la conectividad,

memoria y procesamiento. Toda la comunicación está basada en el intercambio de mensajes. La Plataforma JADE-iPhone 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.

La primera versión del aplicativo JADE-iPhone proporciona una interfaz amigable para el acceso a las funciones básicas del terminal móvil al ejecutar video streaming, permite la creación de agentes y su comunicación mediante mensajes ACL, basado en los estándares FIPA; al ser Iphone una plataforma cerrada y al no existir en el mercado un aplicativo que posibilite la implementación de agentes, el aplicativo permite la transmisión de fotogramas.

Se procedió a 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 el CODEC de video H263, que comprimen mediante algoritmos el tamaño de los fotogramas, para evitar ocupar un gran espacio de memoria en el dispositivo móvil.

El *Proxy* se divide entre el cliente y el servidor, éstos se comunican continuamente para conocer el estado de la transmisión del video streaming, al ocurrir una interrupción el *Proxy* servidor alerta y solicita que se almacenen los fragmentos en el buffer. El manejo de proxies y agentes inteligentes permite distribuir la carga computacional y mejorar la efectividad de la recuperación de la interrupción.

El modelo que realiza la transferencia de video en tiempo real sobre teléfonos móviles iOS, determina las siguientes entidades (Figura 3.12): un Servidor de video RTSP, un *Agente Proxy Servidor* (*APS*), un dispositivo móvil que actuará como cliente, albergando un *Agente Proxy Cliente* (*APC*) y a la aplicación RTSP Cliente la cual se llama Player y viene integrada al dispositivo iPhone, esta genera las peticiones y visualiza el video.

El Servidor se conecta con el APS, este a su vez con el APC mediante una conexión inalámbrica. En los enlaces entre Servidor – APS y APC – Cliente se transmiten mensajes utilizando RTSP que para comunicar video utiliza RTP. Los agentes APS y APC se despliegan en la plataforma JADE y se comunican a través de mensajes FIPA ACL, con lo que se simplifica la comunicación entre agentes evitando declarar ontologías. En JADE-iPhone el APC se divide en dos partes: FrontEnd del APC a ser ejecutado en el teléfono iPhone y el BackEnd en el Servidor, las dos partes se conectan por una red WiFi, donde las posibles desconexiones se gestionan de forma transparente para el APC.

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.

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.

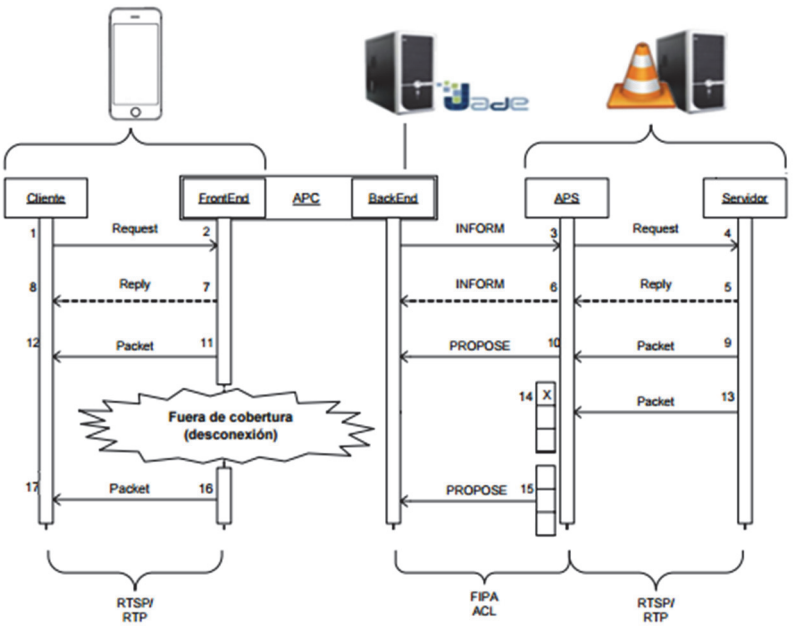

Figura 3.12 Modelo JADE iPhone

Al colocar agentes en el APS y APC, éstos cooperan en definir cuándo el móvil está fuera de cobertura al usar señalización MTP, encargándose de resolver las desconexiones intermitentes y reanudar automáticamente la sesión de video streaming. Al ocurrir una interrupción 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.

Al ocurrir una interrupción inalámbrica, los paquetes que llegan al APS son encapsulados y tratados por el servicio de reparto persistente, este va guardando todos los mensajes que no se pueden entregar a su destinatario en el buffer, hasta cuando el agente APC vuelva a conectarse y se le envíe los mensajes guardados. De esta manera la interrupción por problemas en la red inalámbrica pasa inadvertida para el servidor y el cliente.

Uno de los beneficios en JADE-iPhone 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, éste tiempo influye en el dimensionamiento del buffer. Los fragmentos que no pueden ser visualizados en el cliente porque ha ocurrido una interrupción son guardados en el buffer del cliente. Los agentes inteligentes colaboran en definir el estado de la conexión. Al restablecerse la conexión entre los agentes, el APC lee los frames de vídeo 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.

El código generado posibilita al iOS ejecutar agentes JADE pero con limitantes: no permite ontologías solo frames de tipo *Ordered* y *Qualified* y se debe utilizar el servicio *BEManagementService* en el lado del servidor. Los sockets de comunicación se implantaron en Objective C. Estos sockets empaquetan los mensajes ACL en frames ya sean *Orderer* o *Quailified*, para luego ser convertidos en datos binarios y que se encapsulan en TCP/IP; para poder iniciar la conexión con el servidor JADE, se envían mensajes de HandShacking en la plataforma.

3.3.3. Resultados experimentales

En el escenario de pruebas se utilizó un servidor de video de libre distribución, que es más aceptado a nivel mundial como es VLC. Se ha tratado de emitir el video codificado en un formato compatible con los dispositivos inalámbricos iPhone. La configuración del formato de video es el siguiente: para video el CODEC H.263, tasa de bits de 900 kbps, 25 fps, *ancho*=176, *alto*=144 y para el audio el CODEC MPEG, tasa de bits de 128 kbps, 1 canal y la tasa de muestreo de 44100. Se utilizó una red inalámbrica con tecnología WiFi con un punto de acceso de 50 Mbps, una portátil Intel Core i5 de 2.67Ghz con 8GB de RAM, con tarjeta WiFi (Intel 82577LM) en el cual se ejecuta el servidor de video streaming y el APS; una computadora Intel Core i5 de 1.9 Ghz con 4GB de RAM y tarjeta WiFi (Intel I218-V) en el cual se ejecuta APC backEnd que conecta a un móvil iPhone, en el que se ejecuta el cliente y el frontEnd del APC y un dispositivo móvil iPhone modelo 5S.

En la Figura 3.13 se puede observar una fotografía de este mecanismo en funcionamiento utilizando proxies. En esta imagen, puede verse en la zona inferior el dispositivo móvil y en su pantalla un fotograma del video que el cliente móvil está observando. En la parte central aparece la sucesión de mensajes FIPA ACL que se intercambian entre el APC y el APS, información proporcionada por la utilidad sniffer de la plataforma JADE-iPhone.

Figura 3.13 Funcionalidad de JADE iPhone

Se han realizado varias pruebas, simulando una interrupción en la red, para verificar y comprobar la efectividad de JADE al momento recuperar los paquetes después de una interrupción. Se ha probado que mediante el uso de sistema de agentes JADE-iPhone, las sesiones de video streaming siempre fueron reiniciadas en el punto en que el cliente perdió la conexión.

Para el análisis del jitter y retardo se han analizado 500 frames en dos escenarios, transmisión de video streaming sin proxies y transmisión con proxies. En la figura 3.14 se analiza la variación del jitter, en la cual se observa que el jitter enviando video streaming sin *Proxy* tiene mayores niveles que la transmisión con control de interrupciones.

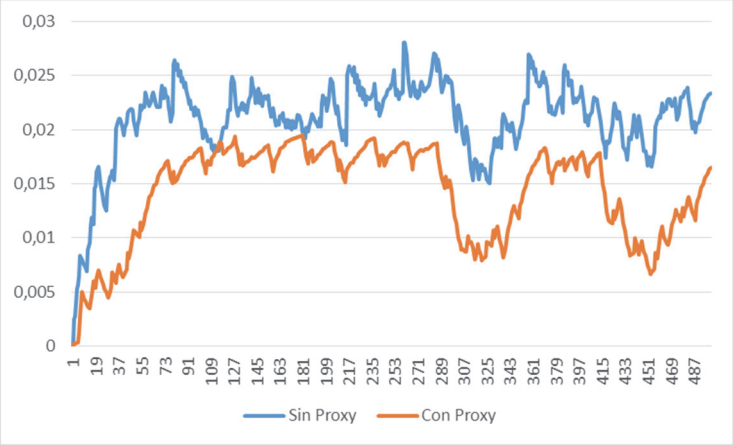

Figura 3.14 Análisis de jitter

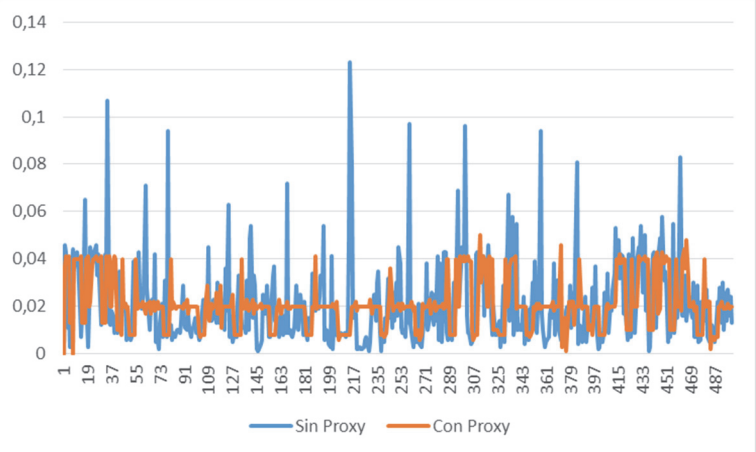

Figura 3.15 Análisis del retardo

El análisis de retardo se presenta en la Figura 3.15 donde puede observarse que el uso de proxies presenta menores retrasos en la transmisión del video streaming.

JADE-iPhone lleva paquetes más grandes ya que posee mayor información como es la de control, pero se puede observar mayor estabilidad en la comunicación, ya que no existe desconexiones, o gran pérdida de paquetes como sucede con RTSP sin Jade iPhone.

4. Solución Web interoperable

Se describe la funcionalidad incrementada a la solución básica con la finalidad de conseguir su despliegue en cualquier tipo de dispositivo o computador, logrando características de interoperabilidad en ambientes Web.

4.1. Análisis del diseño basado en patrones software

En el capítulo anterior se ha presentado la solución básica basada en proxies que ha logrado controlar las interrupciones a través de una arquitectura de proxies, se le ha proporcionado al usuario la continuidad del servicio ya que cuando se ha dado una interrupción ha recibido una notificación de este problema y una vez que el canal se ha restablecido ha continuado visualizando el video desde la última posición de reproducción en la que se encontraba. La implementación de proxies y almacenamiento temporal donde se guardan los flujos mientras se da la interrupción pueden demandar tiempos adicionales de procesamiento e inversión de recursos, pero son totalmente asumibles cuando se ha logrado reducir el impacto de las interrupciones al aminorar los flujos que se retransmiten al darse una interrupción, esto ha impactado positivamente en la mejora de la QoE y QoS.

Esta solución básica ha podido ser implementada experimentalmente en plataformas específicas, como es el caso de iOS, estamos interesados en presentar una solución incremental en el que se establezcan componentes adicionales que permitan lograr la interoperabilidad en ambientes Web.

La solución generada para dispositivos inalámbricos iPhone que controla las desconexiones al transmitir video, solo puede ser aplicada a una plataforma propietaria específica, presentando problemas de flexibilidad y dinamismo al cambio.

Con el avance tecnológico es necesario mejorar la solución con características de interoperabilidad, independencia y multiplataforma. Se desarrolló una arquitectura que permite el control de interrupción interoperable a través de servicios Web independiente del dispositivo cliente utilizado.

A continuación se listan los requisitos funcionales que deben incluirse en el diseño final, los cuales se han establecido mediante diagramas de Casos de Uso y la descripción específica de los mismos. Como puede observarse en la Figura 4.1, en relación al diagrama de caso de uso del modelo básico basado en proxies se mantienen, se ha adicionado un nuevo caso de uso *evaluar estado de recursos del dispositivo*, en el cual se realizan los procesos y actividades tendientes a evaluar ontológicamente el

estado del dispositivo móvil en relación a la batería, latencia y geolocalización para determinar proactivamente la posibilidad de una interrupción. En la tabla 4.1 se establece 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.

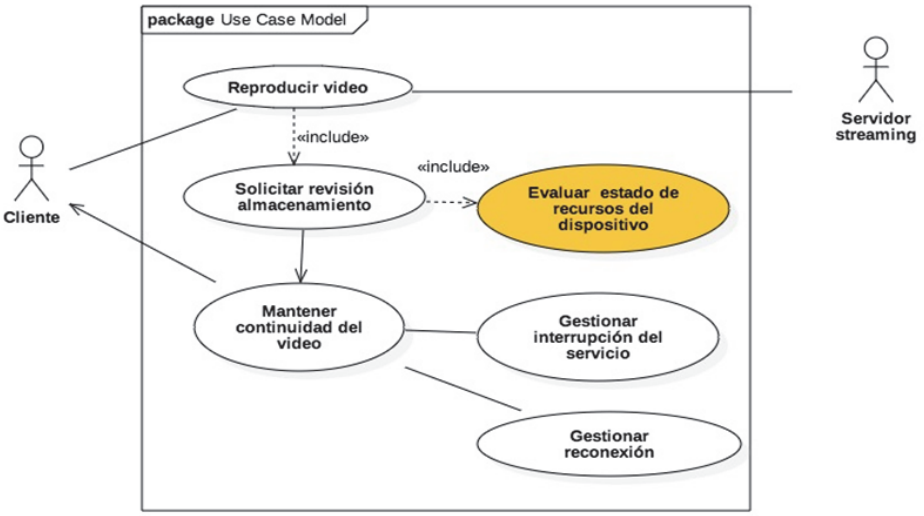

Figura 4.1 Caso de Uso solución interoperable

Tabla 4.1 Caso de uso reproducir video

Caso de Uso: **reproducir video**
Actores: cliente, servidor video streaming
Resumen: se describe puntualmente como inicia la petición de la reproducción del video y su despliegue.
Precondiciones:
Descripción: Curso Normal: 1. El cliente solicita la visualización del video 2. Ingresa los datos del puerto y servidor *3.* Los datos son validados por el *Proxy* 4. Al no existir problemas el *Proxy* solicita el servicio al servidor de video streaming 5. Se despliega una ventana en la que inicia el video streaming 6. El servidor inicia la entrega periódica de los frames al almacenamiento temporal
Pos condiciones: se solicita la revisión continua del almacenamiento
Observaciones:

Se detalla la funcionalidad referente a la revisión periódica del almacenamiento temporal donde se alojan los frames para determinar si hay problemas cuando se supera el umbral permitido (tabla 4.2) y todo el proceso al análisis ontológico de los recursos del dispositivo móvil para establecer si hay riesgo de una interrupción, este caso de uso es posteriormente explotado y definido su detalle (tabla 4.3).

Tabla 4.2 Caso de uso solicitar revisión almacenamiento

Caso de Uso: **solicitar revisión almacenamiento**
Actores:
Resumen: se explica la importancia de verificar el almacenamiento temporal
Precondiciones: el *Proxy* solicita la revisión del almacenamiento
Descripción: Curso Normal: 1. Continuamente se pide la información del estado del almacenamiento 2. Se compara con el nivel máximo establecido 3. Se genera una alerta en caso de superar el umbral definido 4. Se comunica de este problema al *Proxy*
Pos condiciones: se solicita la evaluación del estado de recursos del dispositivo
Observaciones:

Tabla 4.3 Evaluar estado de recursos del dispositivo

Caso de Uso: **evaluar estado de recursos del dispositivo**
Actores:
Resumen: para asegurar que se trata de una interrupción es necesario evaluar ontológicamente el estado de los recursos del dispositivo
Precondiciones: el *Proxy* solicita la evaluación del estado de los recursos
Descripción: Curso Normal: 1. Se realiza el monitoreo del estado de los recursos 2. Se efectúa el análisis ontológico de los recursos: batería, latencia y geolocalización y se determina si existe problema de interrupción 3. Se transforma las acciones del agente a servicios Web 4. Se comunica del problema de interrupción al *Proxy*
Pos condiciones: se solicita mantener la continuidad del servicio
Observaciones:

Se explica el proceso que sigue el *Proxy* para definir la operación a realizar y solventar una interrupción dada (tabla 4.4) y se detalla toda la funcionalidad al ocurrir una interrupción, el uso de buffer para almacenar los frames que han sido enviados por el servidor y que no pueden ser entregados al cliente al haber ocurrido una interrupción (tabla 4.5).

Tabla 4.4 Caso de uso mantener continuidad del video

Caso de Uso: **mantener continuidad del video**
Actores:
Resumen: frente a un evento del canal de comunicación debe gestionarse la continuidad del servicio de video streaming
Precondiciones: el *Proxy* solicita la ejecución de un proceso de acuerdo al estado de los recursos del dispositivo
Descripción: Curso alerta interrupción : 1. En caso de ser una alerta de interrupción se solicita la gestión de interrupción Curso reconexión : 2. Cuando el canal se ha restablecido se solicita la reconexión automática
Pos condiciones:
Observaciones:

Tabla 4.5 Caso de uso gestionar interrupción del servicio

Caso de Uso: **gestionar interrupción del servicio**
Actores: cliente
Resumen: explica el detalle de las actividades a cumplir cuando ocurre una interrupción y como se controla la misma
Precondiciones:
Descripción: Curso normal : 1. Se solicita el almacenamiento de la última posición de reproducción del video 2. Se despliega en la interfaz gráfica del usuario el mensaje que se ha dado un problema y que se encuentra en estado de espera para reconectarse automáticamente 3. Se pide el servicio de evaluar continuamente el canal de comunicación.
Observaciones:

Tabla 4.6 Caso de uso gestionar reconexión

Caso de Uso: **gestionar reconexión**
Actores: Cliente
Resumen: explica el detalle de las actividades a cumplir cuando el canal se ha restablecido
Precondiciones:
Descripción: Curso normal : 1. Se pide al almacenamiento la última posición de reproducción del video 2. El *Proxy* solicita al servidor video streaming la reproducción del video desde la posición especifica 3. El cliente automáticamente continúa visualizando el video en el punto en el que se encontraba cuando ocurrió la interrupción.
Pos condiciones:
Observaciones:

En la tabla 4.6 se explica la recuperación de los frames almacenados temporalmente en el buffer y el despliegue en el cliente una vez que la comunicación se ha restablecido.

De la Figura 4.1 se ha explotado el caso de uso evaluar *estado de recursos del dispositivo* en el que se establece toda la funcionalidad del análisis ontológico del estado de los recursos del dispositivo, los agentes ontológicos ejecutan sus acciones tendientes a establecer si hay una interrupción y su transformación de las acciones del agente a servicios Web interoperables (Figura 4.2).

La fortaleza de esta solución radica en la definición proactiva de la ocurrencia de una interrupción, para lo cual se sustenta en el planteamiento de ontologías que son gestionadas por el agente mediante comportamientos que en conjunto definirán la alerta eminente de una interrupción y la transformación a servicios Web para apoyar en la ejecución de estrategias proactivas que permitan enfrentar la misma.

La nueva funcionalidad de esta solución se muestra claramente en este diagrama de caso de uso en el que se detalla el monitoreo, el análisis ontológico y la consecución de la interoperabilidad a través de servicios Web.

En la tabla 4.7 se muestra las actividades tendientes a realizar el monitoreo constante de los recursos del dispositivo: batería, latencia y geolocalización.

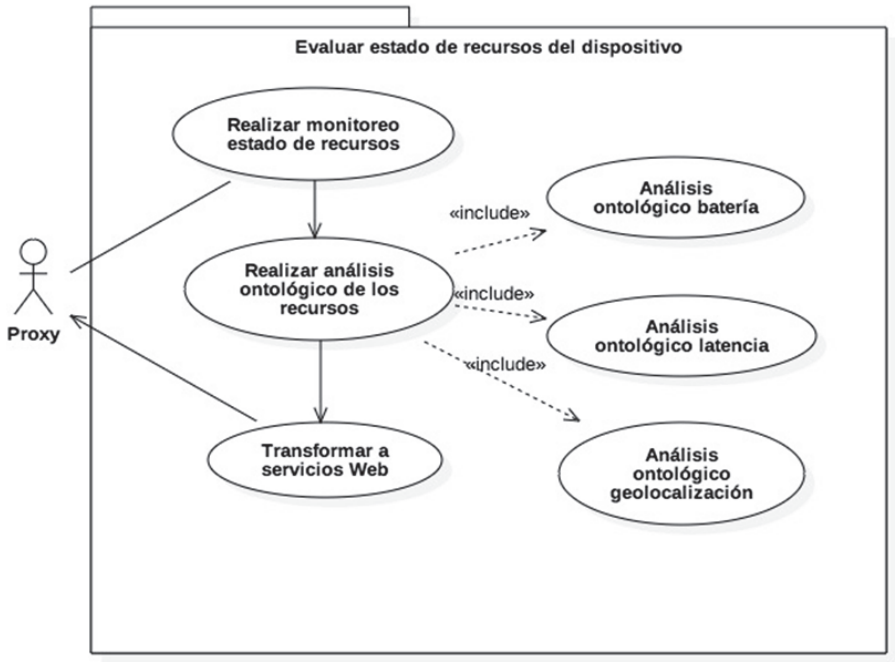

Figura 4.2 Explosión del diagrama del caso de uso

Tabla 4.7 Caso de uso realizar monitoreo estado de recursos

Caso de Uso: **realizar monitoreo estado de recursos**
Actores: *Proxy*
Resumen: se explica la importancia de monitorear el estado de los recursos
Precondiciones: el *Proxy* solicita la revisión del estado de recursos
Descripción: Curso Normal: 1. Continuamente se obtiene la información del estado a los recursos del dispositivo: batería, latencia, geolocalización 2. Esta información es almacenada temporalmente 3. La información de los recursos es monitoreada constantemente 4. Se comunica al *Proxy* el estado de los recursos monitoreados
Pos condiciones: se solicita el análisis ontológico de los recursos
Observaciones:

Se detalla todo el proceso relacionado a la solicitud a realizar el análisis ontológico de: batería, latencia y geolocalización para poder definir si hay una interrupción (tabla 4.8) y la explicación de cómo se establece el análisis ontológico de la batería del dispositivo móvil (tabla 4.9).

Tabla 4.8 Caso de uso realizar análisis ontológico de los recursos

Caso de Uso: **realizar análisis ontológico de los recursos**
Actores:
Resumen: para asegurar que se trata de una interrupción es necesario evaluar ontológicamente el estado de los recursos del dispositivo
Precondiciones: el *Proxy* solicita el análisis ontológico
Descripción: Curso Normal: 1. Se solicita el análisis ontológico de la batería 2. Se requiere el análisis ontológico de la latencia 3. Se requiere el análisis ontológico de la geolocalización 4. Se determina si existe problema de interrupción
Pos condiciones: se solicita transformar a servicios Web
Observaciones:

Tabla 4.9 Caso de uso análisis ontológico batería

Caso de Uso: **análisis ontológico batería**
Actores:
Resumen: se realiza el análisis ontológico de la batería
Precondiciones:
Descripción: Curso Normal: 1. Se revisa el porcentaje de carga de la batería 2. Se establece si el porcentaje está en el rango del 10% al 30% 3. Al estar en ese rango se define riesgo de interrupción derivado de este factor.
Pos condiciones:
Observaciones:

Se detalla todas las actividades tendientes a analizar ontológicamente el estado de la latencia del dispositivo móvil (tabla 4.10) y se explica el proceso del análisis ontológico de la geolocalización (tabla 4.11).

Tabla 4.10 Caso de uso análisis ontológico latencia

Caso de Uso: **análisis ontológico latencia**
Actores:
Resumen: se realiza el análisis ontológico latencia
Precondiciones:
Descripción: Curso Normal: 1. Se revisa la latencia en milisegundos 2. Se establece si esta latencia es superior a los 100 milisegundos 3. Si es superior se establece una alerta de alta probabilidad de interrupción
Pos condiciones:
Observaciones:

Tabla 4.11 Caso de uso realizar análisis ontológico geolocalización

Caso de Uso: **realizar análisis ontológico geolocalización**
Actores:
Resumen: se realiza el análisis ontológico de la geolocalización
Precondiciones:
Descripción: Curso Normal: 1. Se revisa los datos de la latitud y longitud diagnosticando la red 2. Si establece en que rango se encuentra esta información 2.1 0-135 m: conexión favorable 2.2 135-150 m: alerta de conexión en riesgo 2.3 Mayor de 150 m: alerta de interrupción eminente 3. Se define el tipo de alerta de acuerdo al rango explicado
Pos condiciones:
Observaciones:

Tabla 4.12 Caso de uso transformar a servicios web

Caso de Uso: **transformar a servicios Web**
Actores:
Resumen: Se transforman las acciones de los agentes en servicios Web
Precondiciones:
Descripción: Curso Normal: 1. Se obtiene el análisis ontológico de los recursos: batería, latencia y geolocalización y se determina si existe problema de interrupción 2. Se transforma las acciones del agente a servicios Web 3. Se envía los servicios Web al *Proxy*
Pos condiciones:
Observaciones:

En la tabla 4.12 se detalla todo el proceso de transformación de las acciones del agente a servicios web, con la finalidad de indicar al *Proxy* que está ocurriendo una interrupción.

4.1.1. Justificación de los patrones software a utilizar

En la solución básica basado en proxies se planteó un diseño sustentado en patrones software robusto que apoyaron en establecer mecanismos que permitieron gestionar efectivamente el control de una interrupción, este diseño proporciona una organización estructurada del MVC, se potenció a cada una de ellas con los patrones software *Observer*, *Strategy* y *Composite*.

En base a las bondades que ofrece MVC en cuanto a modularidad, flexibilidad, facilidad de cambio, escalabilidad… y a los resultados obtenidos en el capítulo anterior, a la solución se adiciona un observador que se encarga de evaluar y notificar el estado de los recursos monitoreados (batería, latencia y geolocalización) y realizar el análisis ontológico para definir proactivamente si ha ocurrido una interrupción; también se ha incrementado un adaptador que permite adaptar los servicios del agente ontológico en servicios Web para que puedan ser reconocidos en la Web y lograr la interoperabilidad deseada.

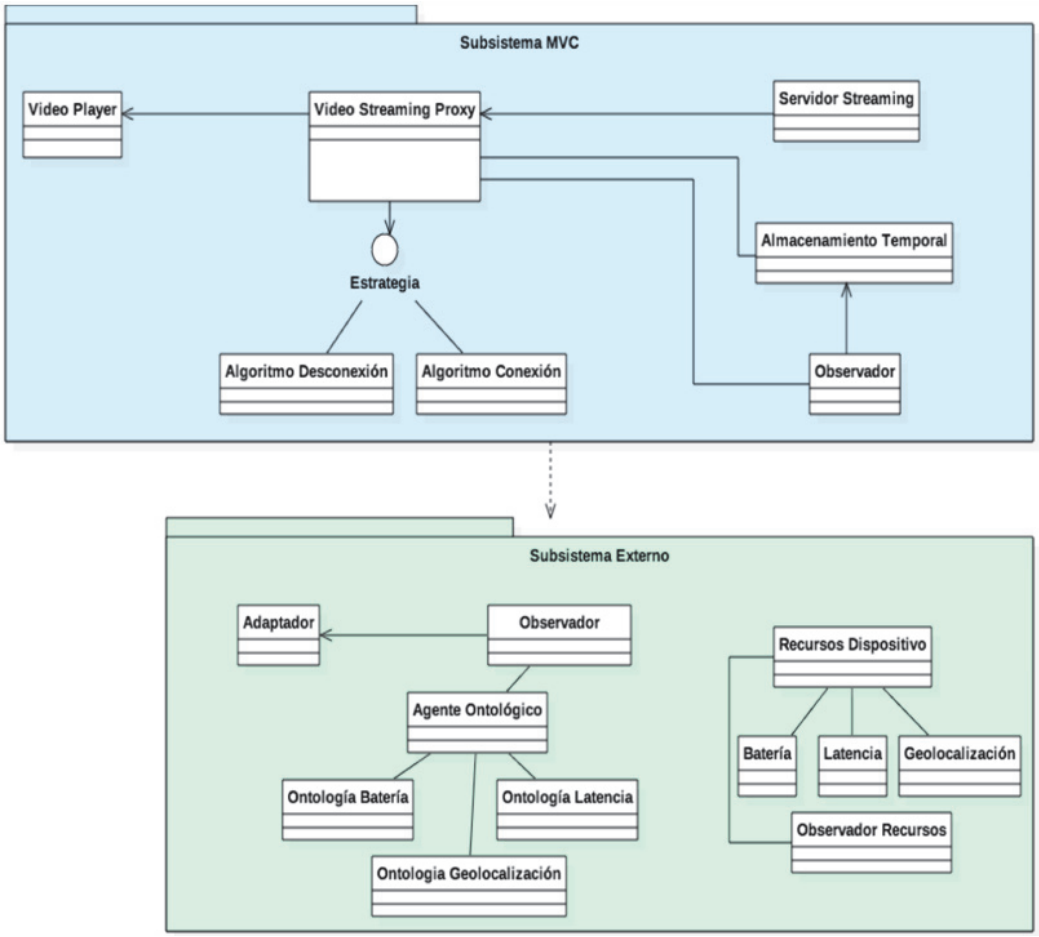

Figura 4.3 Modelo Web interoperable

Como se muestra en 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 ontológico para determinar una interrupción, en este caso se ha utilizado el patrón *Observer*. Para lograr la interoperabilidad, el patrón *Adapter* permite la transformación de las acciones del agente a servicios Web.

A continuación se describe el subsistema MVC:

- *Modelo*: está ubicado en el servidor de aplicaciones, en este se encuentra identificado el almacenamiento temporal, cuando ocurre una interrupción se graba la posición actual de la reproducción del video. La solución se fortalece con la implementación del patrón *Observer* que constantemente está monitoreando el estado del almacenamiento temporal y se lo 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 reproducción del video continúa.
- *Controlador*: está en el servidor de aplicaciones, para complementar su accionar se ha utilizado los patrones software: *Proxy* y *Strategy*. El patrón *Proxy* cumple la labor de intermediario entre el servidor de video streaming y el reproductor de video. Una de sus actividades principales es enviar la información del hardware de dispositivo móvil, al subsistema externo y en caso de recibir la alerta de interrupción solicita al patrón *Strategy* la ejecución del algoritmo de interrupción. Al momento de restablecerse la comunicación, el *Proxy* se encargará de requerir del patrón *Strategy* la ejecución del algoritmo de conexión.

Se describe el subsistema externo:

- En el lado del modelo se almacena la información del hardware del dispositivo cliente: latencia, batería y geolocalización. Este 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 *Observer* externo recibe la información del hardware del dispositivo móvil remitido por el *Proxy*; realiza el análisis ontológico correspondiente y define proactivamente una posible interrupción. En caso de ocurrir este evento el patrón *Adapter* transforma las acciones del agente ontológico a servicios Web interoperables y los envía al *Proxy*.

Es importante detallar la funcionalidad del aplicativo Web interoperable. El reproductor de video solicita el despliegue del video al *Proxy*, éste a su vez lo demanda del servidor de video streaming. Continuamente se registra el estado de los recursos del dispositivo: batería, latencia y geolocalización. El estado de estos recursos es revisado por el patrón *Observer* del subsistema externo y comunicados al *Proxy*.

El *Proxy* solicita al *Observer* del subsistema externo el análisis ontológico del estado de la batería, latencia y geolocalización. Del análisis realizado puede desprenderse la posibilidad de interrupción, en este caso el patrón *Adapter* se encarga de transformar las acciones del agente en servicios Web y los envía al *Proxy* del subsistema MVC. Este evento ocasiona que el *Proxy* solicite al patrón *Strategy* la ejecución del algoritmo de interrupción, el cual demanda que se almacene en el almacenamiento temporal la posición en la que se encuentra el video, despliega en la vista el mensaje de que esta desconectado e inicia el bucle de reconexión.

Cuando el *Proxy* recibe la notificación de que el canal se ha restablecido, solicita al *Strategy* la ejecución del algoritmo de conexión, el cual permite recuperar la información de la última posición del video del almacenamiento temporal y solicita al servidor de video streaming el despliegue del video desde esta posición. En resumen, el aplicativo desarrollado permite el control interoperable de una interrupción; el uso de agentes inteligentes como servicios Web apoyan en la predicción de una interrupción, pudiéndose almacenar la última posición del video, con lo que se logra la continuidad del video streaming.

4.1.2. Diseño arquitectónico y de despliegue

En el diseño se define la arquitectura de la propuesta que soporte los requisitos y dirija a implantar el modelo Web interoperable basado en patrones software. El diagrama de clases del sistema se expone en la Figura 4.4.

A continuación se explica la relación entre el modelo de patrones software y las clases que 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: *ServerStreaming*, *Buffer* y *DataListener*.
- Vista y patrón *Composite*: las clases que son ejecutadas para lograr el despliegue del video streaming en el dispositivo cliente son: *VideoPlayer*, *URLStreaming* y *VideoView*.

- Controlador: en este se define el patrón *Proxy* el cual está representado por la clase *StreamingProxy*. El patrón *Strategy* también fortalece la capa de control implementado mediante las clases: *ActionControler*, *InterrupciónAction* y *ReconexiónAction*.

El observador externo que permite evaluar el estado de los recursos del dispositivo está representado por la clase *RecursosListener*, controla el almacenamiento temporal de las clases *RecursosDispositivo*.

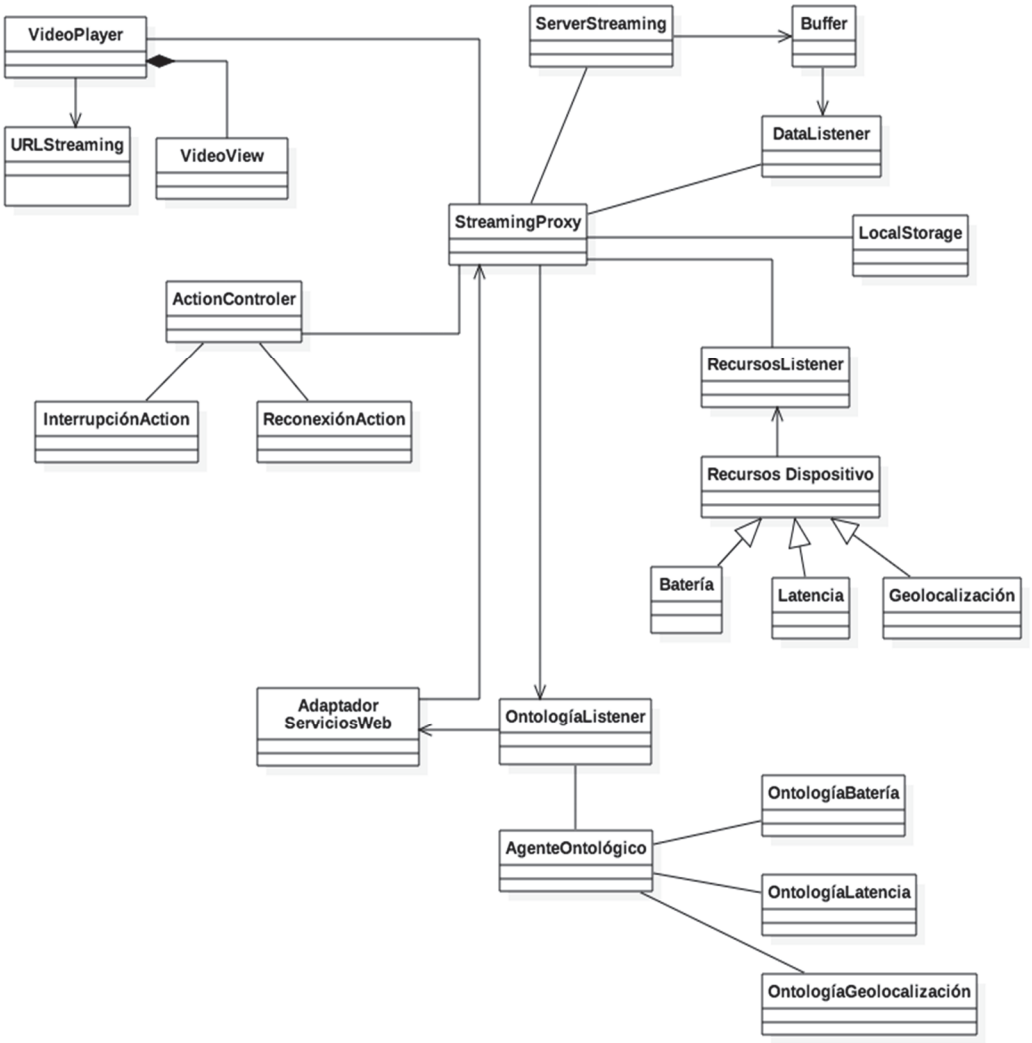

Figura 4.4 Diagrama de clases

El análisis ontológico es solicitado por el observador *OntologiaListener* que solicita a *AgenteOntologico* realizar el análisis de batería, latencia y geolocalización; la respuesta de este análisis es traducido a servicios Web por la clase *AdaptadorServiciosWeb*.

En la Figura 4.5 se muestra el diagrama de secuencia, en la cual se observa la comunicación que se da entre las clases definidas anteriormente para solventar la solicitud y despliegue del servicio de video streaming, una vez establecido este enlace es necesario activar la escucha del control de datos. Si el almacenamiento presenta nivel superior al umbral, este evento es notificado a la capa controladora. Para manejar una interrupción, es importante garantizar su veracidad, por lo que al darse una alerta de almacenamiento, se activa el *Observer* del estado de los recursos del dispositivo y el análisis ontológico correspondiente para determinar una interrupción. Al definir la existencia de una interrupción, se activan las solicitudes necesarias para que la posición de la última reproducción se almacene en el *LocalStorage*. Una vez que el *Observer* indique que el canal se ha restablecido, otras acciones son ejecutadas tendientes a solicitar al servidor de video streaming el envío de los flujos a partir de la posición guardada en el almacenamiento temporal.

En la Figura 4.6 se puede observar el diagrama de despliegue, el dispositivo móvil aloja al *VideoPlayer*, *URL* y *VideoView* que son las clases que inician la solicitud del pedido del servicio de video streaming, esta pedido es recibido por el *AgenteProxyCliente* y enviado al *AgenteProxyServidor* a través de comunicación ACL, esta solicitud es direccionada al *ServerStreaming*, el cual devuelve el flujo hacia el *Proxy*, luego los flujos pasan a ser almacenados en el *Buffer* del dispositivo móvil e inicia su despliegue en el cliente.

Al iniciar esta visualización el *Proxy* solicita el trabajo de observador a *DataListener* quien continuamente está revisando al *Buffer* y define alerta de umbral superado. Si este evento ocurre el *Proxy* solicita el servicio de *RecursosListener* de observar el estado de los recursos del dispositivo móvil (batería, latencia y geolocalización), esta información es enviada al observador *OntologiaListener* que se encarga de evaluar si hay una interrupción mediante el análisis ontológico del agente.

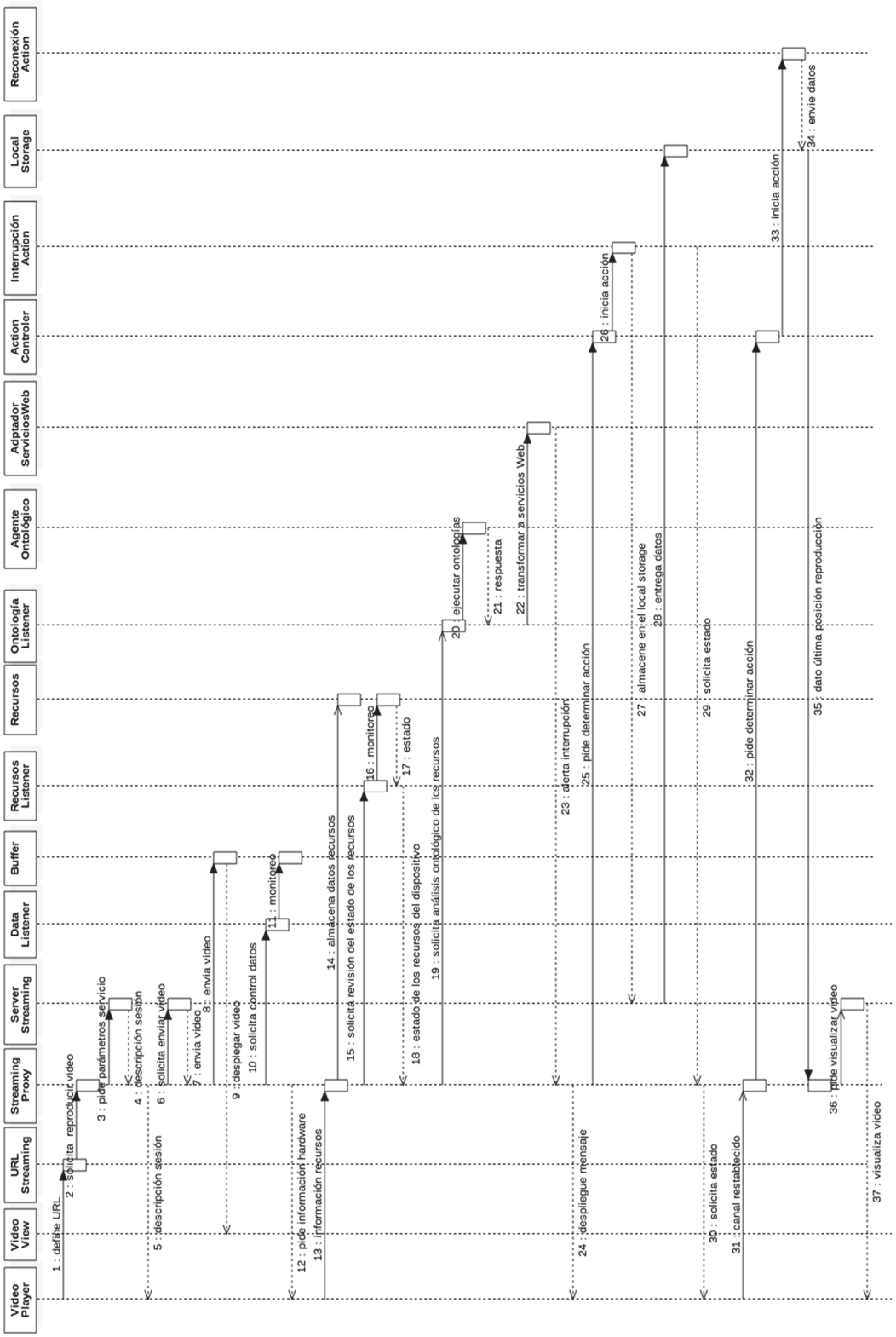

Figura 4.5 Diagrama de secuencia

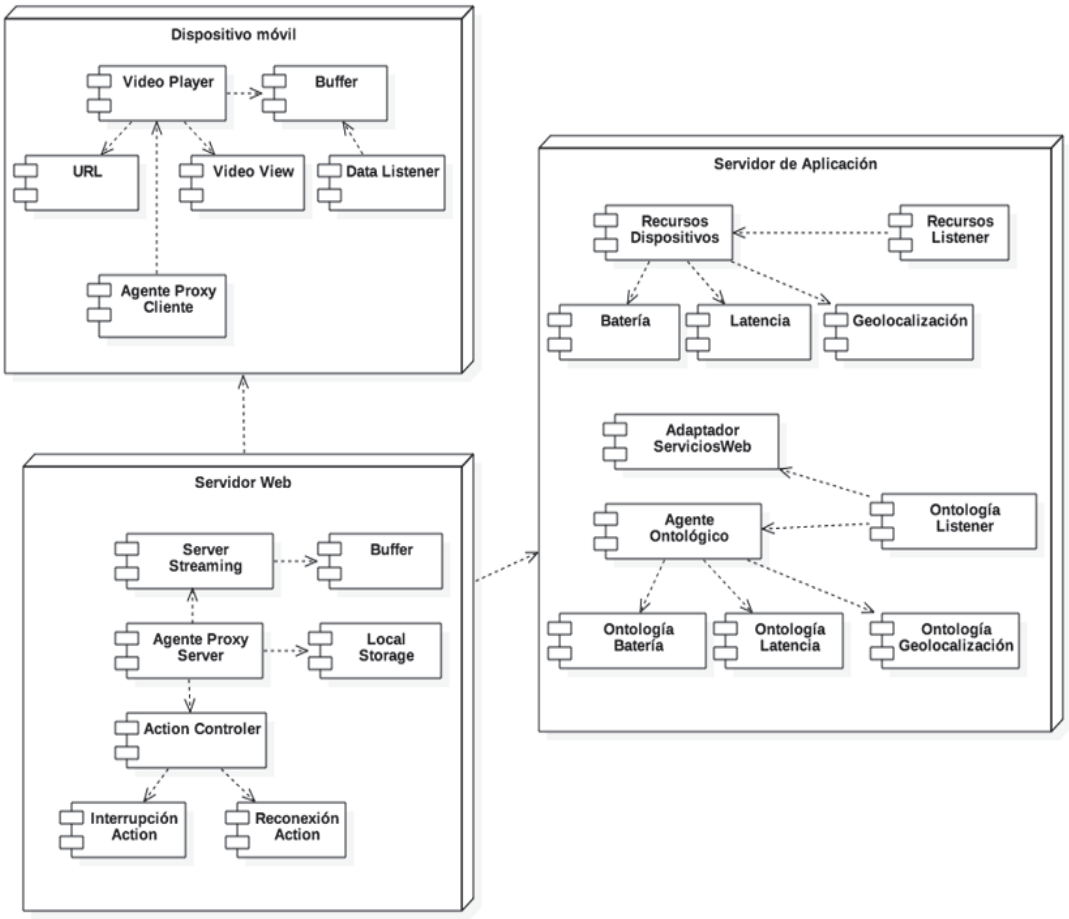

Figura 4.6 Diagrama de despliegue

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 la última posición de reproducción y pide se evalúe el canal y determine su restablecimiento. Al restablecerse el canal el *Proxy* recibe esta notificación y pide al *ActionControler* la ejecución del algoritmo correspondiente, para esta situación sería *ReconexiónAction* mediante el cual el *Proxy* solicita recuperar del almacenamiento temporal la posición grabada y pide al *ServidorStreaming* la transmisión del flujo desde esta posición y sea entregado al proceso de visualización ubicado en el dispositivo móvil.

4.2. Modelo matemático de rendimiento

La solución Web interoperable se basa en la solución básica con proxies, se ha adicionado interoperabilidad a través de agentes ontológicos traducidos a servicios Web. El mecanismo de control de interrupciones ha cambiado ya que el observador ha dejado de monitorear el canal de comunicación, en lugar de ello analiza ontológicamente el estado de los recursos del dispositivo móvil con lo que define proactivamente la ocurrencia de una interrupción. El buffer de almacenamiento, utilizado en la solución con proxies para alojar los frames que no pueden ser enviados al cliente por haberse dado una interrupción del servicio, en el modelo interoperable es utilizado para alojar la última posición del video en la que se encontraba al ocurrir la interrupción.

Basado en lo expuesto el modelo matemático aplicado esta plataforma Web interoperable es la misma que se definió en el capítulo anterior, esto es:

$$T_{exec} = T_{start} \ + \sum_{i=1}^{n} \left(T_{data_i}\right) + \sum_{i=1}^{n} T_{int_i} \ + T_{data_n}$$

Donde:

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

$\sum_{i=1}^{n}\left(T_{data_i}\right)$ es el tiempo total de *Tdata* transmitidos.

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

T_{data_n} es el último frame transmitido.

Para evaluar la potencialidad de la solución Web interoperable se han realizado varias pruebas en vivo con un video de duración de 660,06 segundos y generando 3 interrupciones; estas pruebas se han ejecutado con 1, 2 y 3 usuarios de forma concurrente para determinar si se degrada el uso del canal, se han establecido 8 réplicas por cada número de usuarios, establecidos en la Figura 4.7.

Como puede observarse en la Figura 4.8 se realiza un análisis del tiempo de ejecución de la visualización del video, de forma concurrente por 1, 2 o 3 usuarios, pudiendo denotarse que presenta un comportamiento lineal. En la Figura 4.9, se analiza el total de tiempos de interrupción de 1, 2 y 3 usuarios, la duración de la interrupción se encuentra en el rango de 16,50 a 19,50. Concluyendo que si el video tiene una duración de 660,06 segundos, el tiempo de ejecución es afectado por el tiempo de estas interrupciones.

Replica	#Usuario	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b	c	Texec
										∑(Tdatai)	∑(Tinti)	Tstart + b + c + Tdatan
1	1	0,65	41,60	5,75	47,85	5,45	47,65	5,95	522,96	137,10	17,15	677,86
2	1	1,05	46,15	5,25	47,05	5,45	45,25	5,80	521,61	138,45	16,50	677,61
3	1	1,65	49,10	6,30	43,40	5,35	60,15	6,40	507,41	152,65	18,05	679,76
4	1	0,95	47,60	5,40	40,65	5,80	44,65	5,35	527,16	132,90	16,55	677,56
5	1	0,85	46,20	6,35	45,00	6,40	50,15	6,15	518,72	141,35	18,90	679,81
6	1	1,35	48,00	6,15	45,25	6,25	60,55	5,25	506,26	153,80	17,65	679,06
7	1	0,85	45,20	6,45	53,55	5,70	41,20	5,75	520,11	139,95	17,90	678,81
8	1	1,00	37,15	6,30	51,90	5,75	45,75	5,40	525,26	134,80	17,45	678,51
1	2	1,05	47,90	5,05	49,00	5,70	43,70	6,05	519,46	140,60	16,80	677,91
2	2	0,95	54,00	6,55	46,45	5,35	49,45	5,10	510,16	149,90	17,00	678,01
3	2	1,05	48,60	5,85	46,35	6,50	51,05	5,15	514,06	146,00	17,50	678,61
4	2	1,05	47,20	5,75	43,95	6,20	43,20	5,45	525,71	134,35	17,40	678,51
5	2	1,45	50,20	5,40	46,90	6,05	62,95	8,15	500,01	160,05	19,60	681,11
6	2	0,90	51,35	6,30	46,75	6,30	42,05	5,30	519,91	140,15	17,90	678,86
7	2	0,90	42,35	6,20	47,70	5,30	48,60	5,75	521,41	138,65	17,25	678,21
8	2	1,25	49,80	5,95	41,30	5,85	41,75	5,20	527,21	132,85	17,00	678,31
1	3	0,30	30,70	5,05	37,60	6,55	32,80	5,05	558,96	101,10	16,65	677,01
2	3	0,65	33,60	5,95	43,35	6,00	31,05	5,90	552,06	108,00	17,85	678,56
3	3	1,65	36,65	6,45	33,20	6,35	36,65	6,20	553,56	106,50	19,00	680,71
4	3	0,80	36,45	5,90	31,00	6,25	36,65	5,60	555,96	104,10	17,75	678,61
5	3	1,15	40,95	5,40	29,50	5,95	35,35	6,20	554,26	105,80	17,55	678,76
6	3	1,30	37,30	5,95	44,65	5,70	42,20	5,80	535,91	124,15	17,45	678,81
7	3	0,30	33,80	6,35	34,50	6,15	39,15	5,50	552,61	107,45	18,00	678,36
8	3	1,25	36,95	6,10	36,60	5,90	36,60	6,45	549,91	110,15	18,45	679,76

Figura 4.7 Experimentación tiempo de ejecución

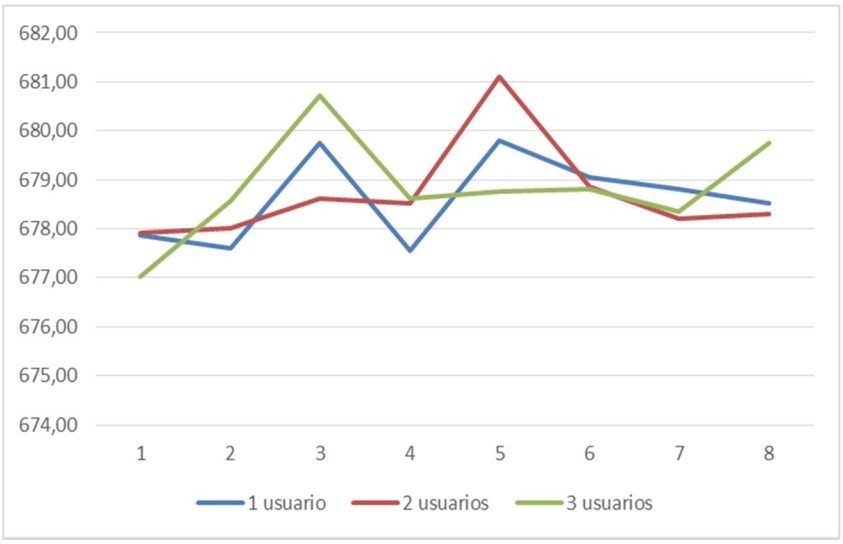

Figura 4.8 Análisis tiempo de ejecución por escenario

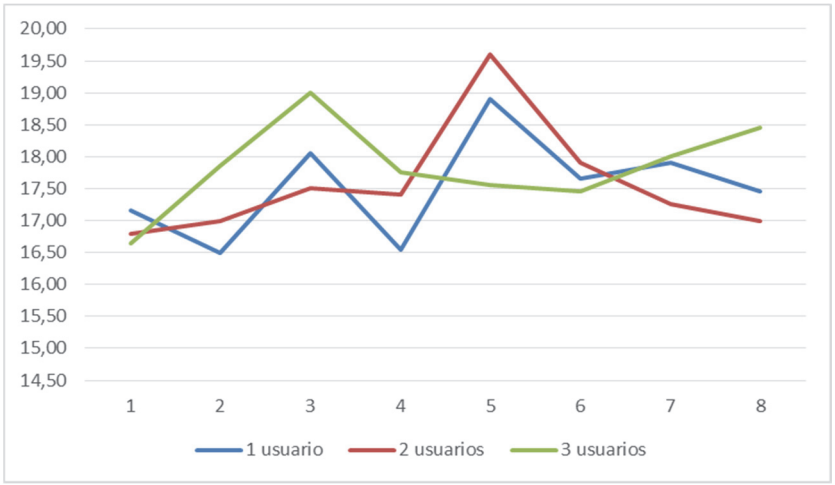

Figura 4.9 Análisis tiempo de interrupción por escenario

Para la ejecución de este modelo se requiere utilizar un almacenamiento temporal de recursos para la gestión ontológica y última posición de reproducción para controlar la interrupción. En la transmisión del video los tiempos de menor duración se han dado por el *Tstart* valor que es tomado en cuenta para valorar el coste de almacenamiento, por lo que $Tstart_1$ =3.

Para evaluar la cantidad de memoria utilizada, se define como velocidad del canal de 10Mbps en ADSL entonces se determina el número de bits que transmite con la siguiente ecuación:

$$Número\ de\ bits = velocidad\ del\ canal * tiempo\ de\ la\ interrupción$$

$$Número\ de\ bits = 3 * 10 = 30\ \ bits$$

Para obtener bytes lo dividimos para 8, obteniendo 30/8 = 3,75 MB

La cantidad de memoria utilizada 3,75 MB para grabar la información de los recursos del dispositivo móvil y la última posición de reproducción del video, es totalmente menor en comparación al almacenamiento requerido en la solución basada en proxies que es de 18,75 MB.

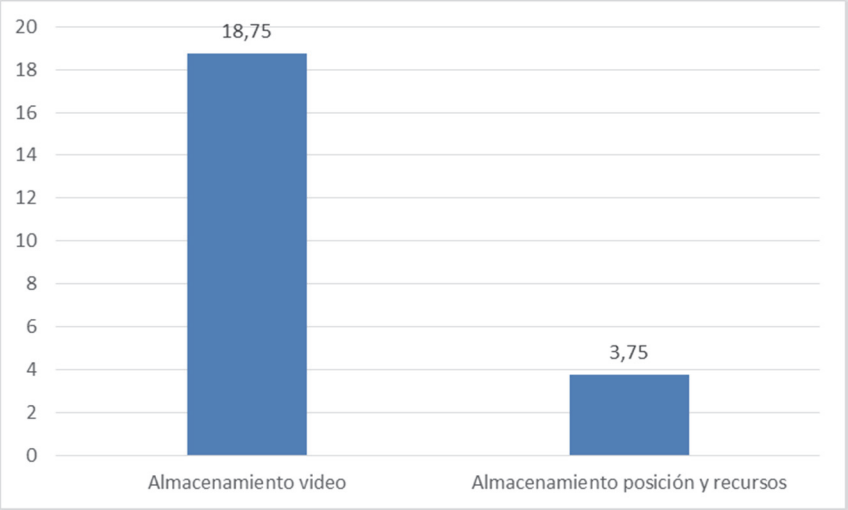

Figura 4.10 Comparativa uso de memoria

En la Figura 4.10. se compara el uso de memoria de la solución web interoperable en relación con la solución básica, concluyendo que es un costo razonable a pagar al recibir los beneficios proporcionados por un mecanismo que controla interrupciones de forma interoperable, aplicado a entornos Web y eliminando las retransmisiones.

4.3. Análisis de la implantación experimental

Se desarrolló la arquitectura denominada *Interrupt Control Architecture of Video streaming by Web Agents* (*ICASWA*) la cual gestiona las interrupciones del video streaming a través del uso de agentes Web. Esta arquitectura está basada en JADE WSIG que transforma los comportamientos del agente inteligente a servicios Web. Los agentes Web interoperables permiten evaluar de manera inteligente el estado de la batería, la latencia y posición geográfica del dispositivo móvil y predecir una eminente interrupción inalámbrica en base a las reglas ontológicas establecidas. Para solventar la interrupción se almacena la última posición de reproducción del video, con la finalidad de tener un punto de partida una vez que la reconexión ocurra

4.3.1. Proyección de diagramas de diseño

La representación en términos de análisis de los aspectos funcionales que deben ser establecidos en la propuesta con traslapados al diseño, para facilitar su orientación hacia el entorno de implementación, en la Figura 4.11 se define el diagrama de clases establecido para esta propuesta.

A continuación se explica la relación entre el modelo de patrones software y las clases que 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: *YouTubeService*, *LocalStorage* y *DataListener*.
- Vista y patrón *Composite*: las clases que son ejecutadas para lograr el despliegue del video streaming en el dispositivo cliente son: *VideoPlayer*, *URL* y *VideoView*.
- Controlador: en este se define el patrón *Proxy* el cual está representado por la clase *ControlServer*. El patrón *Strategy* también fortalece la capa de control implementado mediante las clases: *ActionControl*, *InterruptionControl* y *ConnectionControl*.

El observador externo que permite evaluar el estado de los recursos del dispositivo está representado por la clase *ResourcesListener*, controla el almacenamiento temporal *LocalStorage*. En el subsistema externo se define el análisis ontológico a través de las clases *WiFiAgent*, *WiFiOntology*, *Vocabulary*, *Latency*, *Battery*, *Distance*; la clase encargada de representar al patrón *Adapter* es *WebServiceControl*.

El servicio de video streaming proporcionado por YouTube y controlado a través de ICASWA, es desplegado en el dispositivo cliente a través del browser, cuando se predice una eminente interrupción gracias al análisis ontológico de los recursos del dispositivo móvil, dos actividades suceden: el usuario continúa visualizando los frames almacenados en el buffer del cliente y se graba la posición última del video en el local storage del browser cliente.

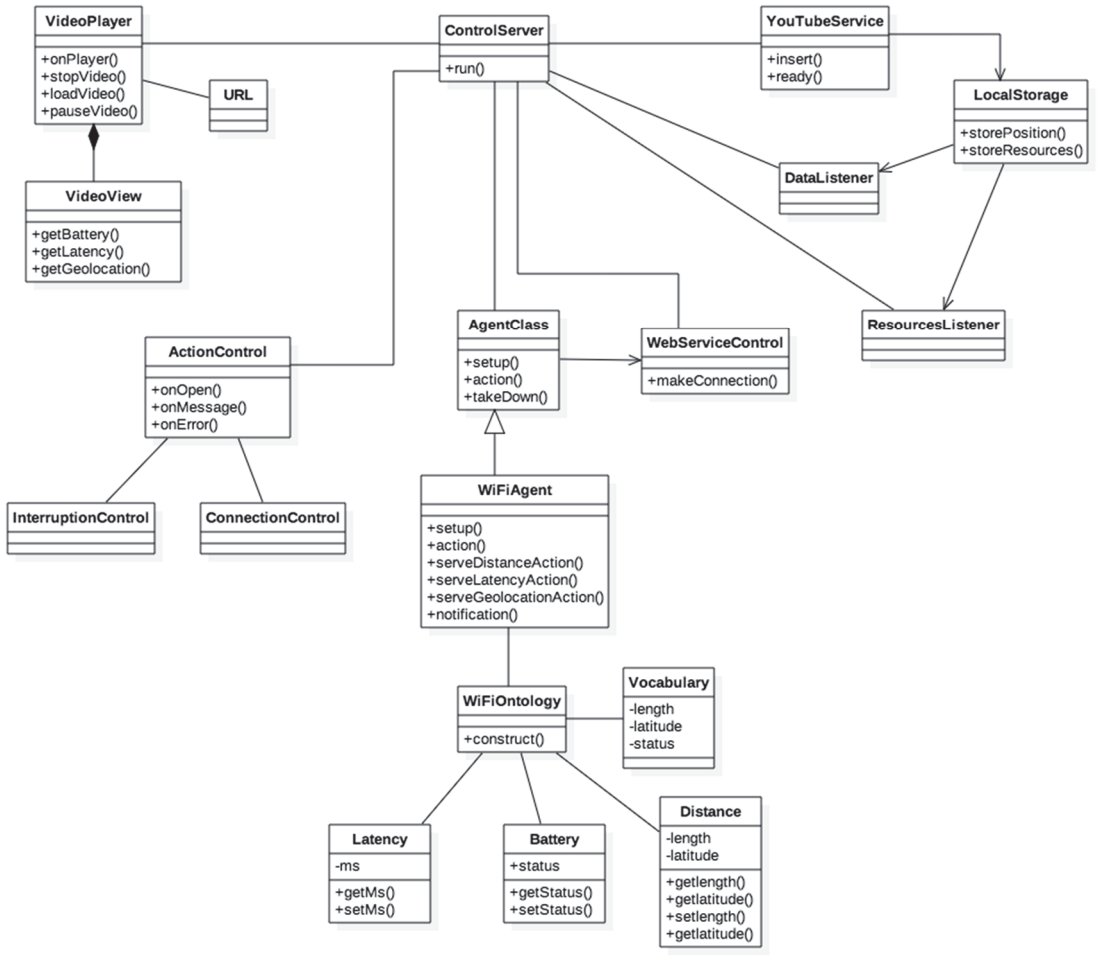

Figura 4.11 Diagrama de clases

Cuando el cliente deja de recibir frames y el servicio se detiene, ICASWA entra en un bucle de reconexión de tal manera que al momento de regresar la cobertura, solicita el servicio de video streaming desde la última posición almacenada y enlazará con el Web socket. El video se despliega en el cliente desde la posición grabada en el local storage. El diagrama de secuencia se muestra en la Figura 4.12.

En la Figura 4.13 se puede observar el diagrama de despliegue, el dispositivo móvil aloja al *VideoPlayer*, *URL* y *VideoView* que son las clases que inician la solicitud del pedido del servicio de video streaming, esta pedido es recibido por el proxy *ControlServer* y direccionada a *YouTubeService* servidor de video streaming, el cual devuelve el flujo hacia el *Proxy*, luego los flujos pasan a ser almacenados en el *LocalStorage* del dispositivo móvil e inicia su despliegue en el cliente.

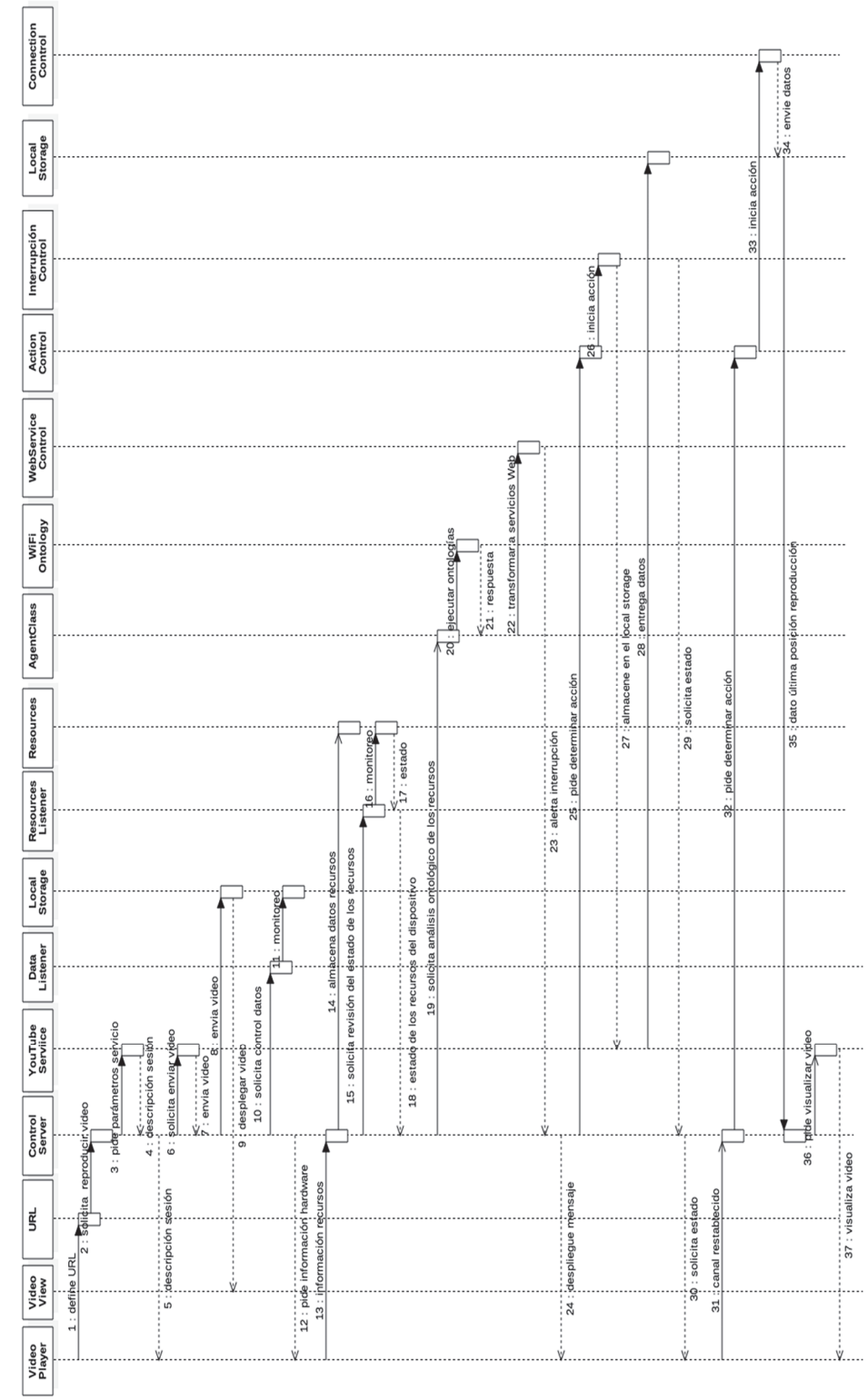

Figura 4.12 Diagrama de secuencia

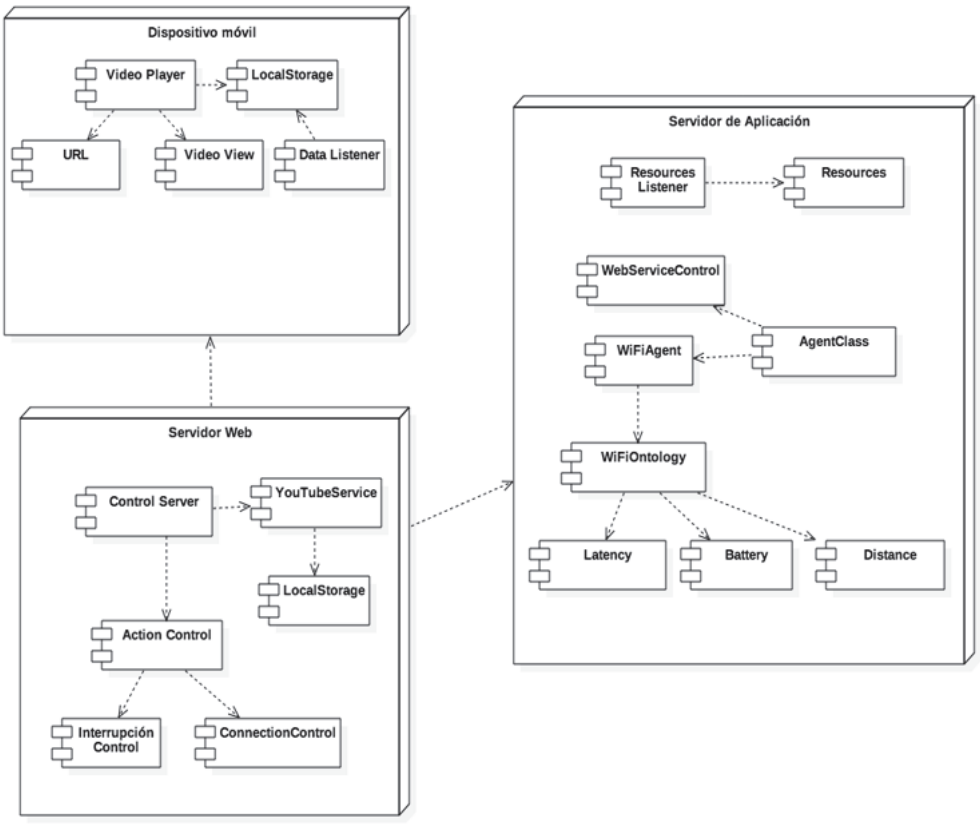

Figura 4.13 Diagrama de despliegue

Al iniciar esta visualización el *Proxy* solicita el trabajo de observador a *DataListener* quien continuamente está revisando al *LocalStorage* y define alerta de umbral superado. Si este evento ocurre el *Proxy* solicita el servicio a *ResourcesListener* de observar el estado de los recursos del dispositivo móvil (batería, latencia y geolocalización), esta información es enviada al observador *AgentClass* que se encarga de evaluar si hay una interrupción mediante el análisis ontológico del agente. Si esta respuesta asegura que hay una interrupción el *Proxy* invoca al servicio del *ActionControl* para que seleccione el algoritmo a ejecutar, en este caso *InterruptionControl*.

Este algoritmo se encarga de que el *Proxy* solicite al *YouTubeService* el almacenamiento temporal de la última posición de reproducción en el *LocalStorage* y pide se evalúe el canal y determine su restablecimiento. Al restablecerse el canal el *Proxy* recibe esta notificación y pide al *ActionControl* la ejecución del algoritmo

correspondiente, para esta situación sería *ConnectionControl* mediante el cual el *Proxy* solicita recuperar del almacenamiento temporal la posición grabada y pide al *YouTubeService* la transmisión del flujo desde esta posición y sea entregado al proceso de visualización ubicado en el dispositivo móvil.

4.3.2. Desarrollo del modelo con software libre

En el desarrollo de la arquitectura ICASWA (Figura 4.14) se utilizó como servidor de video streaming a YouTube, que proporciona una API (YouTube API) basada en JavaScript [156], la cual permite gestionar un video y realizar acciones como: pausar, adelantar, retroceder o reproducir desde una posición específica, ésta última es de gran utilidad cuando ocurre una interrupción. La YouTube API se encuentra implementada en una página Web, basada en HTML5 [157] que se encarga de proporcionar la información del hardware del dispositivo cliente (latencia, geolocalización y batería) al JADE por medio del Web socket. Al suscitarse una interrupción, los datos del hardware y la posición actual de reproducción del video, se almacenan proactivamente en la cache del cliente utilizando la API localStorage. Cuando se reanuda la comunicación, se envía al servidor YouTube la solicitud de retransmisión del video desde la última posición de reproducción almacenada.

Al ser necesario conocer constantemente el estado del dispositivo cliente, se utilizó un servidor de Web sockets (Ratchet WebSockets) en PHP [158] alojado en el puerto 443, el cual establece un canal único de comunicación entre los servicios Web proporcionados por JADE WSIG y el cliente, independiente del servicio de video streaming; cada 1,5 segundos el dispositivo cliente es censado y sus datos son procesados en JADE, obteniendo una respuesta satisfactoria o una alerta que permite almacenar la posición actual del video cuando ocurre una interrupción. Como medio de interacción y comunicación se utilizó el servidor de aplicaciones Apache Tompcat versión 5.5 alojado en el puerto 8080, donde se levantan los servicios Web del WiFiAgent por medio de JADE WSIG. JADE únicamente se encarga del diagnóstico de la red, utilizando WiFiAgent, que posee tres comportamientos para el análisis del

dispositivo los cuales son: estado de la batería, latencia con el servidor y geo localización del punto de acceso más cercano.

Se establece el análisis de la evolución de las API de JavaScript y HTML5 que apoyan al desarrollo de aplicaciones Web, al permitir obtener información de los recursos del hardware del dispositivo móvil; basado en este informe y las pruebas realizadas se ha seleccionado las APIs de:

- Geolocalización: se utiliza para obtener la posición geográfica del usuario, la geolocalización es mucho más precisa para dispositivos con GPS (teléfonos inteligentes) como el iPhone y Android.
- Latencia: mide la suma de los retrasos temporales dentro de la red producido por la demora de transmisión de paquetes en milisegundos.
- Estado de la batería: proporciona información sobre el nivel de carga de la batería y permite notificar eventos cuando hay un cambio en el nivel de la misma.

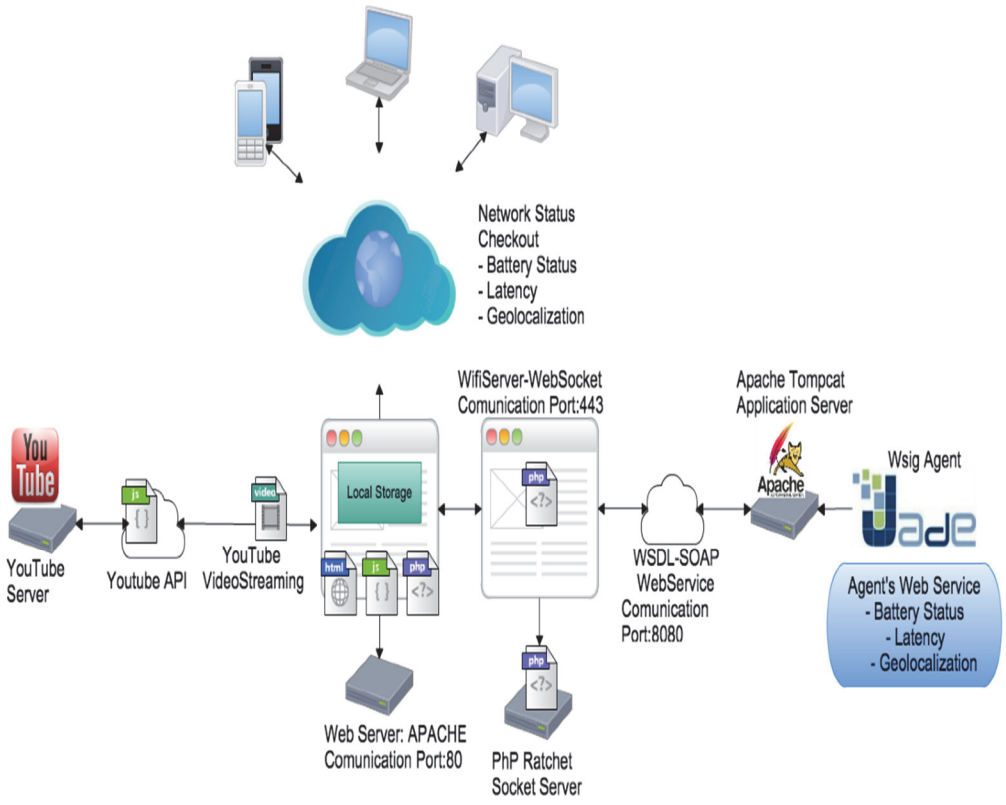

Figura 4.14 Plataforma de servicios Web y agentes inteligentes

Tabla 4.13 Descripción de la ontología WiFiOntology

ONTOLOGIA: **WiFiOntology**					
CONCEPT		**PREDICATE**		**ACTION**	
Batería	Int: Status	BateriaAgent	BATERIA: Bateria	serveDistanciaAction	BATERIA: Bateria
Distancia	Float: Latitud Float: Longitud	DistanciaAgent	DISTANCIA: Distancia	serveDistanciaAction	DISTANCIA: Distancia
Latencia	Int: ms	LatenciaAgent	LATENCIA: Latencia	serveLatenciaAction	LATENCIA: Latencia

Para el almacenamiento de la última posición de reproducción del video al suscitarse una interrupción se utilizó la API:

- Local storage API: permite almacenar datos en el navegador, en lugar de llamar al servidor.

Las API antes mencionadas son productos probados y estables liberados para su utilización, a la fecha se encuentran en proceso de desarrollo y validación otras API para HTML5 desarrolladas en JavaScript que permiten la gestión de otros recursos vía Web. De entre todos los navegadores existentes en el mercado, tanto los de distribución libre como pagada: Google Chrome, Mozilla Firefox, Opera, Safari, Internet Explorer, únicamente tres navegadores, de toda esta lista, presentan un buen soporte con HTML5. Dichos navegadores son Firefox, Chrome y Safari, siendo Firefox el único de los tres que soporta todas y cada una de las API que contemplan esta propuesta.

En el ambiente JADE se creó el agente WiFiAgent, el cual presenta tres comportamientos: estado de la batería, latencia y geolocalización, en éste agente se define la ontología (WiFiOntology) correspondiente al control de estado del dispositivo, que se detalla en la Tabla 4.13.

De acuerdo a la Figura 4.15, para la implementación de WiFiAgent y la creación de los servicios dentro del mapeo ontológico se modificaron las siguientes clases:

- *Ontology*: definición de las instancias de los servicios (procedimientos) del agente.
- *Vocabulary*: definición de los nuevos datos a ser almacenados.

- *Agent Class*: se declara el conjunto de acciones que realiza el agente y el servicio Web posteriormente.
- *Action Class*: corresponde al manejador del servicio ontológico del agente, sus atributos son los definidos en el *Vocabulary*.

Los comportamientos que ofrece WiFiAgent son:

- Battery status: recibe el porcentaje de carga de la batería y si éste se encuentra en un rango del 10% al 30% existe riesgo de interrupción derivado de este factor.
- Latency: recibe la medición de latencia en milisegundos, si ésta es mayor a los 100 ms, el servicio devuelve una alerta de alta probabilidad de interrupción.
- Geolocalization: recibe los datos de la latitud y longitud diagnosticando la red según los siguientes intervalos:
    - 0-135 m: conexión favorable.
    - 135-150 m: alerta de conexión en riesgo.
    - mayor de 150 m: alerta de interrupción eminente.

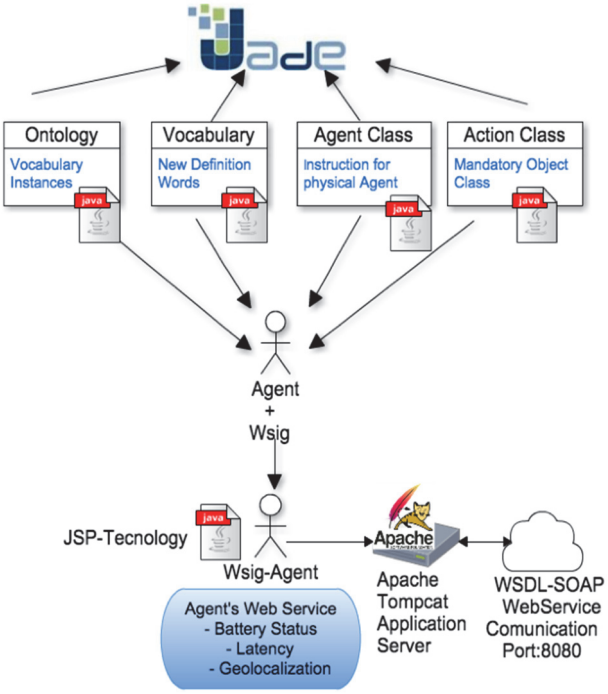

Figura 4.15 Esquema de la Ontología

Los agentes inteligentes al ser transformados en servicios Web, proporcionan nuevas características como la interoperabilidad y multiplataforma, que posibilita la interacción y ejecución de la información y procesos sin importar la plataforma tecnológica sobre la que han sido generados. El desarrollo de aplicaciones Web basados en agentes inteligentes interoperables se benefician de las propiedades de autonomía, pro actividad y dinamismo aportadas por los agentes inteligentes y de la interoperabilidad de datos y procesos proporcionadas por lo servicios Web. En la tabla 4.14 se detalla las áreas y los estándares a cumplirse para proporcionar los comportamientos de los agentes como servicios Web.

Una vez creado el WiFi Agent en la plataforma JADE, al levantar el servidor Glassfish o Tomcat se crea automáticamente el agente WSIG, mostrado en la Figura 4.16.

De acuerdo a la Figura 4.17, WSIG presenta una interface gráfica sencilla que permite revisar los servicios Web controlados por la ontología. En este caso se levantaron tres servicios Web

Tabla 4.14 Integración de agentes inteligentes como servicios Web

AREA	AGENTES	SERVICIOS WEB
Descripción del servicio	AF Agent Description	WSDL
Registro	DF/ AMS	UDDI
Comunicación	ACL	SOAP

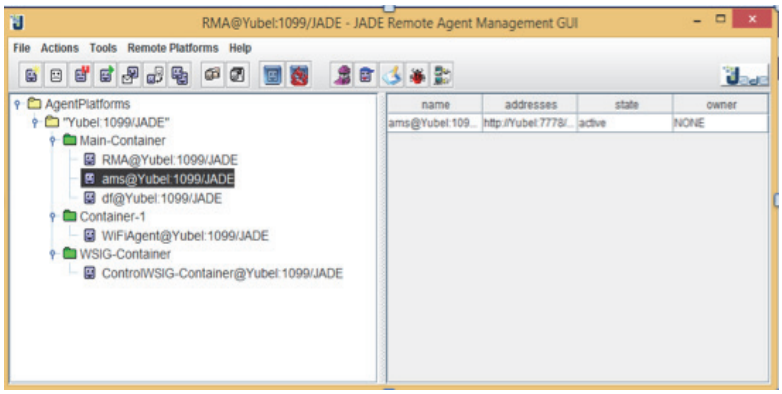

Figura 4.16 Creación agente WSIG

La consola WSIG ofrece un tester para probar el funcionamiento del servicio Web (Figura 4.18), en la sección SOAP Request registra el mensaje de solicitud de servicio XML y en la sección SOAP Response generará los mensajes de respuesta XML del servicio solicitado.

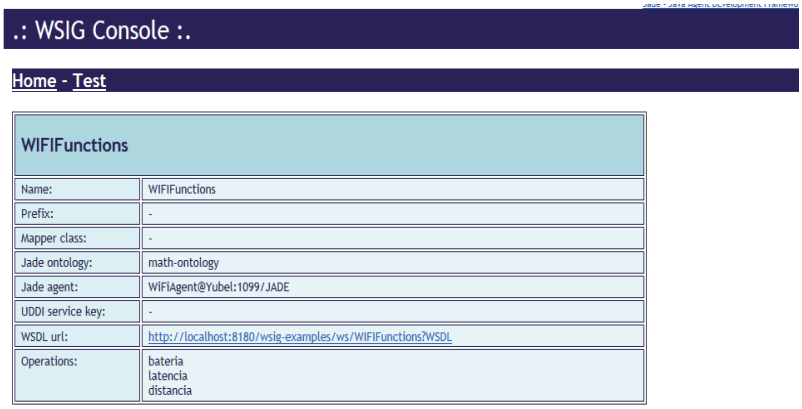

Figura 4.17 Consola WSIG

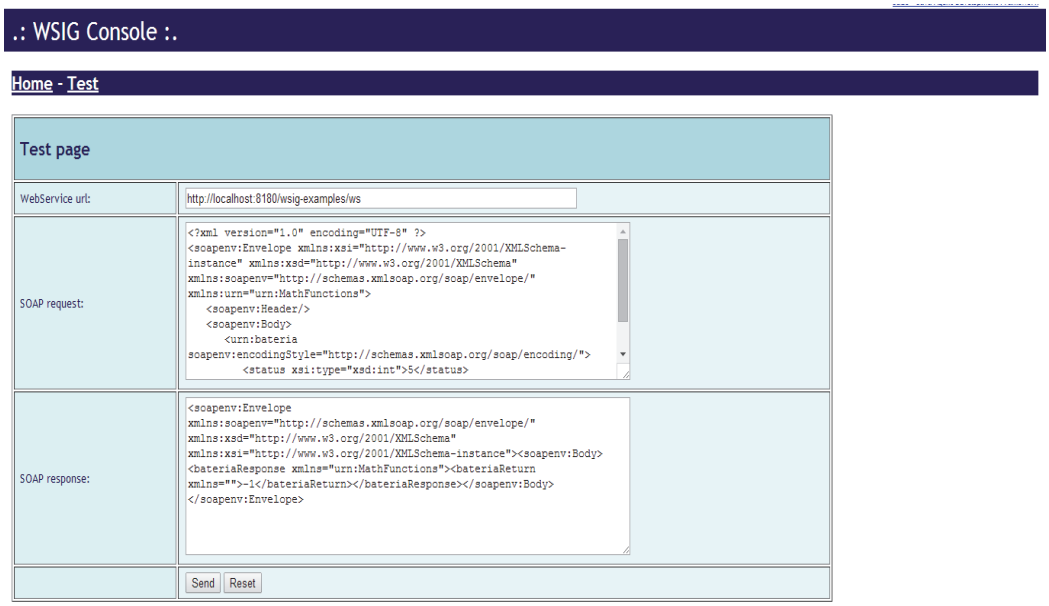

Figura 4.18 Tester WSIG

4.3.3. Resultados experimentales

La arquitectura desarrollada reside en un servidor virtualizado con 4 núcleos de procesamiento, memoria RAM de 8GB y sistema operativo Linux. Se utilizó una Mac Book Pro con procesador Intel core I7 de 2.0 GHz, 4GB en RAM y Sistema Operativo OS X Yosemite V 10.10.3, una red WiFi con velocidad de enlace promedio de 150 Mbps, una latencia promedio de 22ms y un poder de señal promedio de -38dBm, un celular Samsung Galaxy Note 2 con procesador Samsung Exynos 4 Quad 4412/ARM Cortex.A9, 1,6Ghz, 2GB de RAM y Sistema operativo Android 4.1.2 Jelly Bean.

Se trabajó con el servidor de video streaming de YouTube, para la transmisión de video utiliza el estándar H.264 [159], [160], [161] que define un CODEC de alta compresión y regula la calidad de la imagen con tasas binarias notablemente inferiores al estándar previo (H.263). En el caso de audio, el formato utilizado *es Advanced Audio Coding* (*ACC*) que es un algoritmo de codificación de banda ancha de audio que tiene un rendimiento superior al del MP3, ya que produce una mejor calidad y requiere menos recursos del sistema para codificar y descodificar.

Figura 4.19 Tiempos en experimentación en vivo

Para evaluar la efectividad del modelo matemático se ha planteado varias pruebas reales con un video de duración de 660,06 con 3 interrupciones, en dispositivos móviles Samsung Galaxy Note 2, con navegador Firefox, Figuras 4.19 y 4.20.

En la Figura 4.21 se indican los tiempos tomados en las experimentaciones en vivo, concluyendo que los datos sometidos al modelo matemático son resultados casi cercanos a los reales.

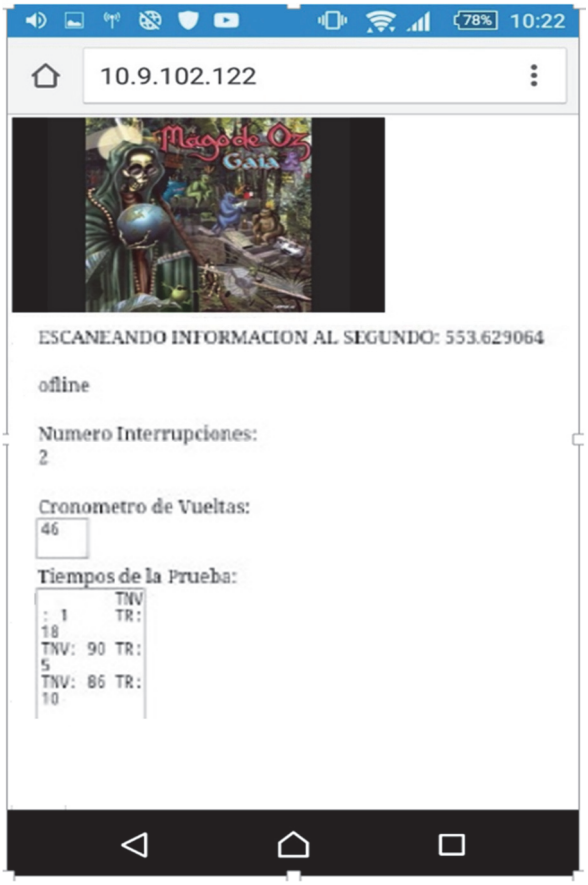

Figura 4.20 Obtención de información en vivo

Prueba	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b	c	Texec
									∑(Tdatai)	∑(Tinti)	Tstart + b + c + Tdatan
1	3,10	30,20	9,60	42,60	9,00	40,50	9,03	554,80	113,30	27,63	698,83
2	1,58	44,60	12,23	42,71	13,13	55,26	11,91	523,46	142,57	37,27	704,88
3	1,88	37,94	12,79	31,45	13,74	37,99	12,46	556,20	107,38	38,99	704,45

Figura 4.21 Datos resultantes de la prueba simulada

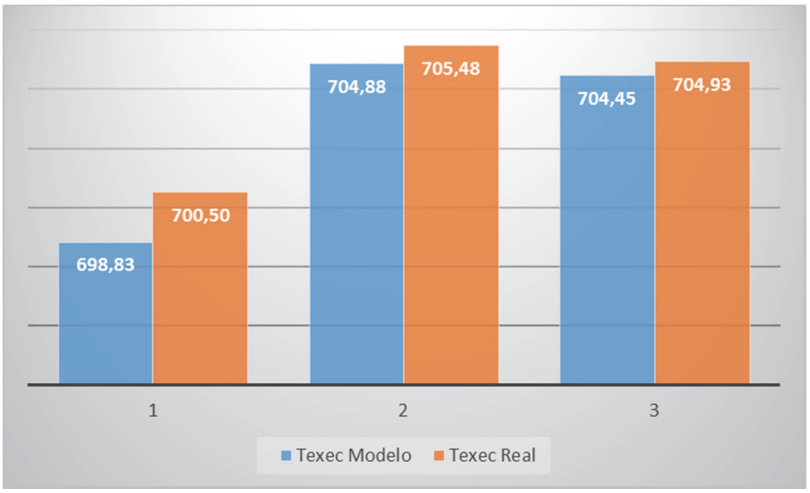

Figura 4.22 Comparativa tiempo de ejecución

En la figura 4.22 se muestra las diferencias entre el tiempo de ejecución real y experimental, concluyendo la efectividad del modelo matemático definido ya que la diferencia de cálculo no supera el 1%.

Se realizaron otras pruebas en vivo en las que se buscó evaluar la afectación de la concurrencia con 1, 2 y 3 usuarios.

Para el escenario de pruebas se establecieron los siguientes parámetros:

- Tiempo de duración del video: 660,06 segundos.
- tiempo de reproducción en línea antes de interrupción manual: 10 segundos.
- número de escenarios: 18 (de 1 a 6 interrupciones) y con 1,2 y 3 usuarios concurrentemente reproduciendo el video.
- número de pruebas por escenario: 30 réplicas.

En la figura 4.23 se resume el promedio de las pruebas realizadas por 1, 2 y 3 usuarios a la vez.

Las pruebas de 2 y 3 usuarios concurrentes se realizaron en horas de la mañana donde la congestión de la red inalámbrica era normal, razón por lo cual en la Figura 4.24 hay un comportamiento uniforme en las mismas. Las pruebas de 1 usuario se efectuaron en la tarde donde los eventos de mayor presencia de estudiantes y ejecución de

procesos de la red han congestionando la red, por lo que presenta mayores tiempos de ejecución en comparación con las pruebas realizadas con 2 y 3 usuarios.

También se realizaron pruebas por 6 ocasiones con diferentes número de interrupciones (de 1 a 6) las cuales han sido ejecutadas con y sin la arquitectura ICASWA; en la tabla 4.15 se establece el valor promedio de las 6 pruebas realizadas por cada interrupción.

En la Figura 4.25 se evalúan las pruebas reales de la arquitectura, los valores promedio sin ICASWA presenta una curva creciente en el tiempo, la curva que representa los valores promedio con ICASWA, muestra una tendencia lineal que oscila entre los 630 y 730 segundos, lo que permite visualizar que la arquitectura propuesta elimina el tiempo de retransmisión y ofrece el servicio de reconexión automático, influyendo de manera positiva en el tiempo de ejecución del video streaming, constituyéndose en un influyente positivo en la QoE del usuario.

usuarios	Tstart	Tdata1	Tint1	Tstart2	Tdata2	Tint2	Tstart3	Tdata3	Tint3	Tstart4	Tdata4
3	1,11	39,03	4,43	3,81	43,13	3,89	3,84	42,97	3,87	3,88	534,93
2	0,54	42,71	5,74	2,01	41,74	5,87	1,82	42,40	5,77	1,84	533,22
1	4,10	48,02	9,69	4,96	47,86	9,90	5,39	45,70	10,08	5,02	518,49

Figura 4.23 Datos promedios prueba con 3 interrupciones

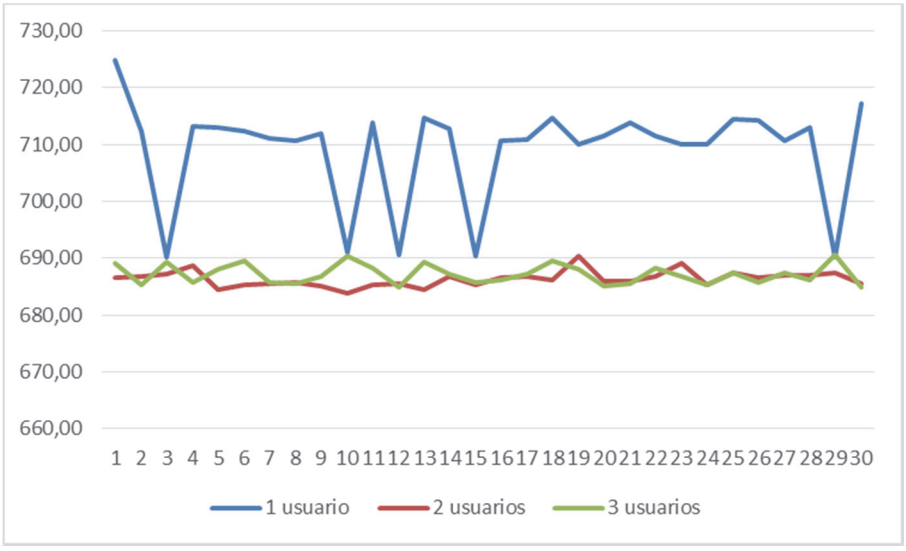

Figura 4.24 Tiempo ejecución con 3 interrupciones

Tabla 4.15 Pruebas reales Arquitectura ICASWA

Número de Interrupciones	Valor sin ICASWA	Valor con ICASWA
1	724,43	676,44
2	779,02	684,78
3	834,65	697,49
4	891,73	707,75
5	967,21	719,46
6	1016,46	732,25

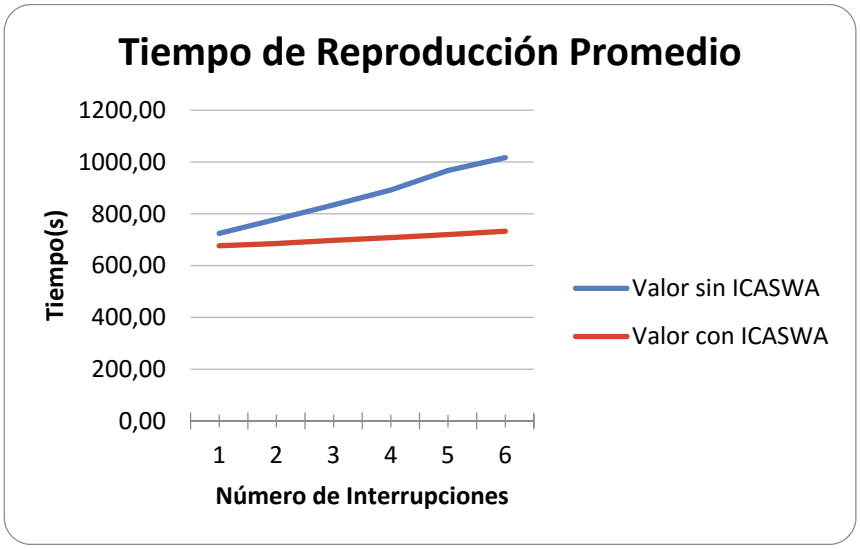

Figura 4.25 Pruebas reales con la arquitectura ICASWA

En las pruebas realizadas se utilizó Mozilla Firefox tanto en celulares con el sistema operativo Android y computadores de escritorio. Dichas pruebas pudieron ser realizadas en la plataforma Mac, únicamente en portátiles que tienen instalado el navegador Mozilla Firefox, estas pruebas no se pudieron realizar en los dispositivos inalámbricos de Apple, por existir incompatibilidad con las APIs utilizadas en ICASWA.

Con miras a medir la efectividad de la arquitectura ICASWA, se desarrolló un código de simulación para ejecutar 1000 pruebas de forma automática, para lo cual se analizó los resultados obtenidos en las pruebas reales, estableciendo los rangos de los parámetros planteados en el modelo matemático.

Se utilizaron tablas de frecuencia con 6 clases, definidas en función del mínimo y el máximo experimental; se seleccionó las clases que presentaron mayor número de incidencias, eliminando aquellas no representativas.

En las tablas 4.16 y 4.17 se observan los rangos elegidos para cada parámetro del modelo matemático. Es necesario recalcar que en el escenario de pruebas, la unidad de tiempo para las mediciones fue segundos (s), tomando en cuenta las pruebas realizadas.

En la tabla 4.18, se establecen los resultados obtenidos en las pruebas simuladas ejecutadas con y sin la arquitectura ICASWA; se ha definido el valor promedio de las 1000 pruebas realizadas por cada interrupción.

De acuerdo a la Figura 4.26, en las pruebas simuladas, la tendencia de los valores obtenidos en el ambiente real se mantiene, pero, al ser la gráfica producto de 1000 pruebas, la tendencia por parte de ambos gráficos es uniforme, manteniendo la misma dirección, proporcionalidad y orientación. De las pruebas realizadas, se deduce que el modelo matemático fue correctamente construido en función del ambiente real, ya que, la simulación muestra resultados que no varían con la realidad y corroboran los beneficios proporcionados a la QoE al reducir el impacto negativo de las retransmisiones.

Tabla 4.16 Rangos sin ICASWA

Variable	MIN	MAX
Tstart	2,01	2,87
Tdata	46,525	56,12
Tint	3,67	4,97

Tabla 4.17 Rangos con ICASWA

Variable	MIN	MAX
Tstart	0,60	0,88
Tdata	38,38	47,07
Tint	10,23	13,45

Los resultados experimentales demuestran que un video streaming controlado por agentes inteligentes como servicios Web, puede mejorar su eficiencia, ya que el análisis oportuno del dispositivo cliente en la predicción de una interrupción, permite la continuidad del servicio video streaming. Los tiempos de dispersión y variabilidad obtenidos, nos han permitido demostrar que el tiempo de reproducción de un video conforme al número de interrupciones es menor utilizando la arquitectura ICASWA.

Esta arquitectura, establece ventajas como son la interoperabilidad y multiplataforma al permitir la aplicación de agentes inteligentes como servicios Web.

Tabla 4.18 Pruebas simuladas

Número de Interrupciones	Valor Sin ICASWA Promedio	Valor Con ICASWA Promedio
1	717,76	671,08
2	775,86	682,20
3	833,34	693,23
4	890,73	704,17
5	949,02	715,13
6	1006,35	726,54

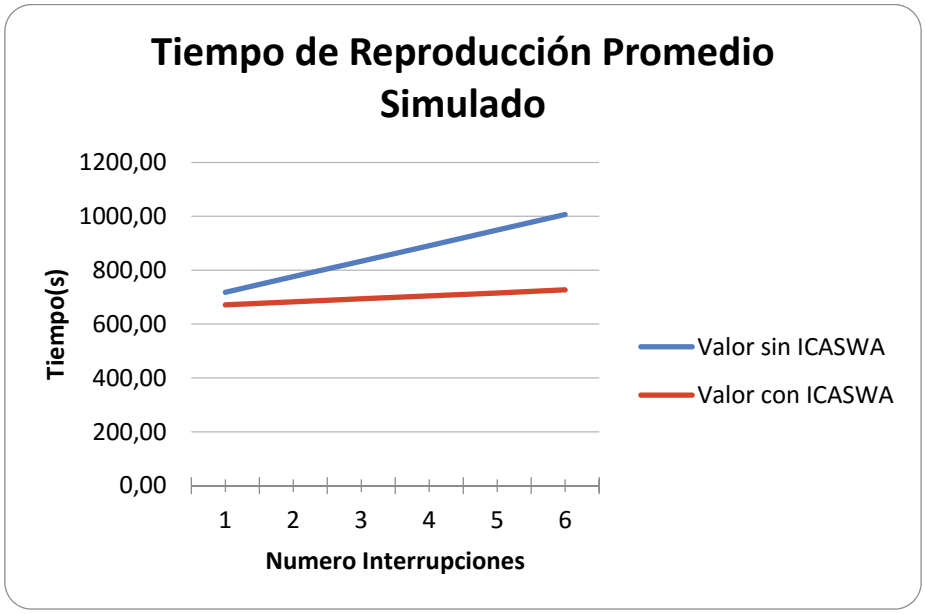

Figura 4.26 Pruebas simuladas de la arquitectura ICASWA

Los agentes desarrollados, presentan un comportamiento ontológico inteligente para predecir una interrupción inalámbrica y actuar proactivamente al guardarse la última posición del video. Generalmente las interrupciones de servicio de video streaming, provocan que el usuario tenga que reiniciar su reproducción, ICASWA soluciona esta problemática mediante un bucle de reconexión que censa constantemente el canal, una vez que se reanuda la comunicación, éste bucle solicita el servicio video streaming desde la última posición almacenada, manteniendo la continuidad de la reproducción.

Se ha probado que ICASWA, soluciona las interrupciones de la transmisión de video streaming en ambientes inalámbricos, aminorando el impacto negativo que ocasiona una interrupción de video en el usuario, ya que reduce la pérdida de tiempo que tendría el cliente en la reproducción del video con interrupciones y evita el abandono de la sesión.

5. Solución para video en tiempo real basada en sensores y servicios Web

Se explica a detalle el diseño e implementación de funcionalidad adicional a la solución Web interoperable, establecido para ser aplicado en ambientes domóticos que utilizan sensores, para activar la reproducción de video en tiempo real en dispositivos inalámbricos y lograr el control de interrupciones.

5.1. Análisis del diseño basado en patrones software

En el capítulo anterior se ha presentado el modelo Web interoperable que ha logrado controlar las interrupciones a través de una arquitectura de proxies y servicios Web, se le ha proporcionado al usuario la continuidad del servicio ya que cuando se ha dado una interrupción ha recibido una notificación de este problema y una vez que el canal se ha restablecido ha continuado visualizando el video desde la última posición de reproducción en la que se encontraba. Con la implementación de este modelo interoperable aplicado a ambientes Web se ha mantenido la fortaleza de los proxies al controlar la interrupción, añadiendo valores agregados como son la interoperabilidad y multiplataforma.

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 ontológicos que notifican la posibilidad de una interrupción y permiten definir acciones proactivas para enfrentar este problema, se ha logrado mejorar la QoE.

Estamos interesados en mantener la arquitectura de software interoperable para implantar el mecanismo de control de interrupciones en la transmisión de video en tiempo real, el cual se dispara en base a la acción de los sensores los cuales notifican una alerta de seguridad.

La continua miniaturización de los dispositivos, el aumento de la capacidad de computación y los avances en las técnicas de reducción de consumo de energía permiten la aparición de nuevos modelos de recopilación de información, como los sensores, éstos son capaces de integrarse con las redes de datos de una manera rápida y transparente.

La introducción de ontologías en las redes de sensores permite la construcción de ambientes semánticos, que analizan el entorno y toman decisiones. Una de estas acciones es la captura de lo que está sucediendo en el hábitat controlado, el análisis del estado de los sensores, la alerta de seguridad y la transmisión del video streaming en tiempo real al dispositivo móvil del cliente.

Es necesario identificar los requisitos funcionales que deben incluirse en el diseño, los cuales se han establecido mediante diagramas de Casos de Uso (Figura 5.1). Cabe indicar que se mantiene la funcionalidad del modelo básico basado en proxies, en el que se realiza la observación del almacenamiento temporal para determinar si este supera el umbral establecido, en este caso se efectúa la revisión del canal de comunicación para establecer que se trata de una interrupción, activando al *Proxy* el cual coordina las actividades necesarias para controlar una interrupción y también gestionar la reconexión automática cuando se restablezca el canal de comunicación.

Se ha establecido incrementos detallados en el caso de uso *evaluar estado de sensores*, donde el agente ontológico evalúa el ambiente controlado y establece si se ha dado una alerta de seguridad, en este caso se activa el video streaming en tiempo real (tabla 5.1); se expone todas 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 (tabla 5.2).

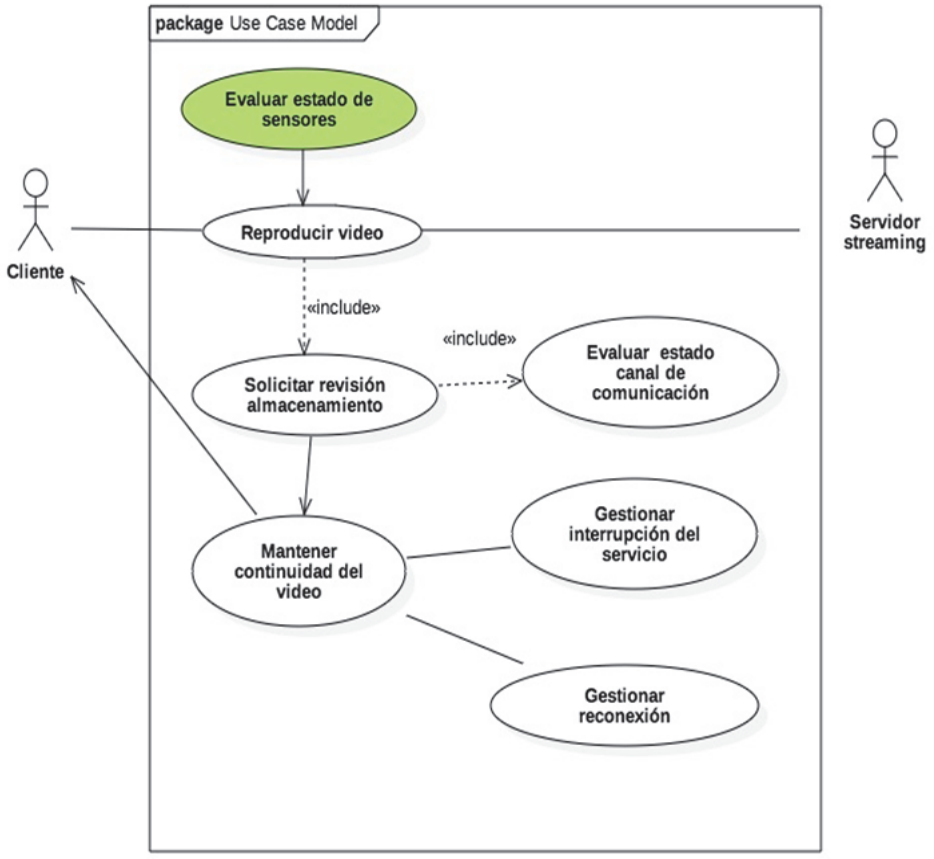

Figura 5.1 Diagrama de caso de uso

Tabla 5.1 Caso de uso evaluar estado de sensores

Caso de Uso: **evaluar estado de sensores**
Actores:
Resumen: para controlar el ambiente es necesario conocer el estado de los sensores y definir la alerta de seguridad en caso de intrusos.
Precondiciones: El ambiente ha sido cambiado a modo seguro
Descripción: Curso Normal: 1. Se realiza el monitoreo del estado de los sensores 2. Se efectúa el análisis ontológico de los recursos: temperatura, luminosidad y movimiento y se determina si existe problema de interrupción 3. Se transforma las acciones del agente a servicios Web 4. Se comunica del problema de alerta de seguridad al Proxy
Pos condiciones:
Observaciones:

Tabla 5.2 Caso de uso reproducir video

Caso de Uso: **reproducir video**
Actores: cliente, servidor video streaming
Resumen: se describe puntualmente como inicia la petición de la reproducción del video y su despliegue.
Precondiciones: *Proxy* recibe alerta de seguridad
Descripción: Curso Normal: 1. El *Proxy* solicita la activación de la cámara 2. Se solicita la visualización del video streaming en tiempo real en el dispositivo cliente 3. Se despliega una ventana en la que inicia el video streaming 4. El servidor de video inicia la entrega periódica de los frames al almacenamiento temporal
Pos condiciones: se solicita la revisión continua del almacenamiento
Observaciones:

Se detalla la funcionalidad referente a la revisión periódica del almacenamiento temporal donde se alojan los frames para determinar si hay problemas cuando se supera el umbral permitido (tabla 5.3) y todo el proceso relacionado con evaluar el estado del canal de comunicación inalámbrico y establecer si se ha dado una interrupción (tabla 5.4).

Tabla 5.3 Caso de uso solicitar revisión almacenamiento

Caso de Uso: **solicitar revisión almacenamiento**
Actores:
Resumen: se explica la importancia de verificar el almacenamiento temporal
Precondiciones: el *Proxy* solicita la revisión del almacenamiento
Descripción: Curso Normal: 1. Continuamente se pide la información del estado del almacenamiento 2. Se compara con el nivel máximo establecido 3. Se genera una alerta en caso de superar el umbral definido 4. Se comunica de este problema al *Proxy*
Pos condiciones:
Observaciones:

Tabla 5.4 Caso de uso evaluar estado canal de comunicación

Caso de Uso: **evaluar estado canal de comunicación**
Actores:
Resumen: para asegurar que se trata de una interrupción es necesario evaluar cómo está la comunicación inalámbrica.
Precondiciones: el *Proxy* solicita la evaluación del canal de comunicación
Descripción: Curso Normal: 1. Se evalúa la comunicación a nivel de RTSP 2. Se evalúa la comunicación a nivel de UDP 3. Se determina el estado del canal de comunicación 4. Este estado es comunicado al *Proxy*
Pos condiciones: se solicita mantener la continuidad del servicio
Observaciones:

Se explica el proceso que sigue el *Proxy* para definir la operación a realizar y solventar una interrupción dada (tabla 5.5) y se detalla toda la funcionalidad al ocurrir una interrupción, en este caso todos los frames capturados por la cámara son guardados en un almacenamiento temporal mientras dure una interrupción, cuya duración tiene un tiempo límite para no saturar el espacio de almacenamiento (tabla 5.6).

Tabla 5.5 Caso de uso mantener continuidad del video

Caso de Uso: **mantener continuidad del video**
Actores:
Resumen: frente a un evento del canal de comunicación debe gestionarse la continuidad del servicio de video streaming
Precondiciones: el *Proxy* solicita la ejecución de un proceso de acuerdo al estado de los recursos del dispositivo
Descripción: Curso alerta interrupción : 1. En caso de ser una alerta de interrupción se solicita la gestión de interrupción Curso reconexión : 2. Cuando el canal se ha restablecido se solicita la reconexión automática
Observaciones:

Tabla 5.6 Caso de uso gestionar interrupción del servicio

Caso de Uso: **gestionar interrupción del servicio**
Actores: cliente
Resumen: explica el detalle de las actividades a cumplir cuando ocurre una interrupción y como se controla la misma
Precondiciones:
Descripción: Curso normal : 1. Se solicita el almacenamiento de lo que está capturando la cámara mientras dure la interrupción 2. Se despliega en la interfaz gráfica del usuario el mensaje que se ha dado un problema y que se encuentra en estado de espera para reconectarse automáticamente 3. Se pide el servicio de evaluar continuamente el canal de comunicación.
Observaciones:

En la tabla 5.7 se explica la recuperación de la interrupción y el despliegue del video en el cliente una vez que la comunicación se ha restablecido. De la Figura 5.1 se ha explotado el caso de uso *evaluar estado de sensores* en el que se establece toda la funcionalidad del análisis ontológico del estado de los sensores y su transformación de las acciones del agente a servicios Web interoperables (Figura 5.2).

Tabla 5.7 Caso de uso gestionar reconexión

Caso de Uso: **gestionar reconexión**
Actores: Cliente
Resumen: explica el detalle de las actividades a cumplir cuando el canal se ha restablecido
Precondiciones:
Descripción: Curso normal : 1. El *Proxy* solicita el envío de la información a la cámara de video 2. El cliente automáticamente continúa visualizando el video 3. El cliente tiene la opción de reproducir el video almacenado en caso de requerir verificar que ocurrió mientras se dio la interrupción.
Pos condiciones:
Observaciones:

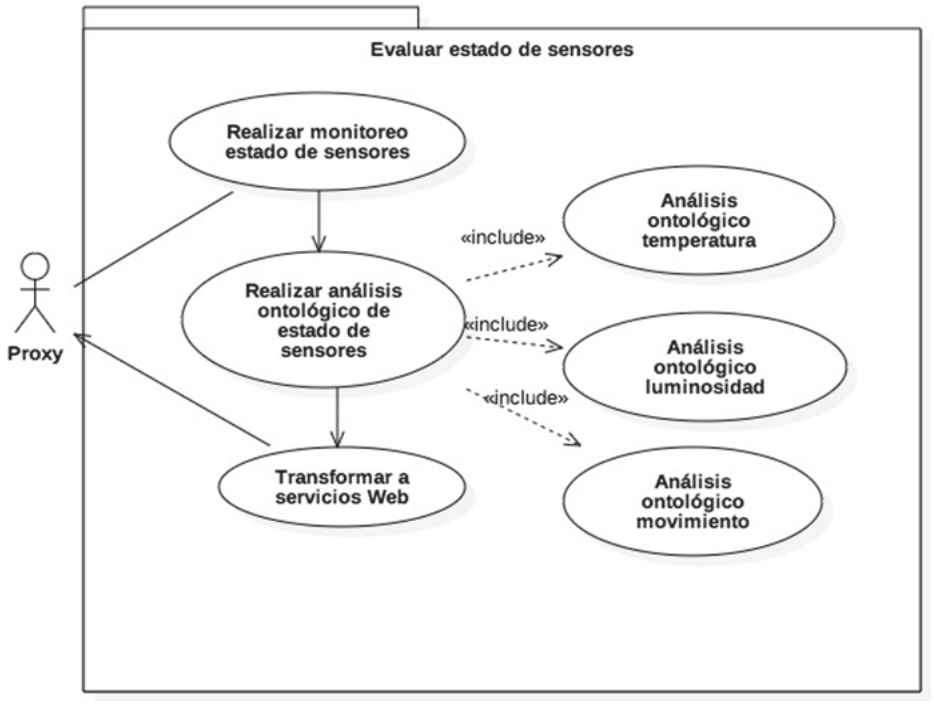

Figura 5.2 Explosión del caso de uso evaluar estado de sensores

Se explican todas las actividades tendientes a realizar el monitoreo constante del estado de los sensores (tabla 5.8) y todo el proceso para solicitar el análisis ontológico de: temperatura, luminosidad y movimiento para poder definir si hay una alerta de seguridad (tabla 5.9).

Tabla 5.8 Realizar monitoreo estado de recursos

Caso de Uso: **realizar monitoreo estado de recursos**
Actores: *Proxy*
Resumen: se explica la importancia de monitorear el estado de los sensores
Precondiciones: el *Proxy* solicita la revisión del estado de sensores
Descripción: Curso Normal: 1. Continuamente se obtiene la información del estado de los sensores: temperatura, luminosidad y movimiento 2. Esta información es almacenada temporalmente 3. La información de los sensores es monitoreada constantemente 4. Se comunica al *Proxy* el estado de los sensores monitoreados
Pos condiciones: se solicita el análisis ontológico de los recursos
Observaciones:

Tabla 5.9 Caso de uso realizar análisis ontológico de los sensores

Caso de Uso: **realizar análisis ontológico de los sensores**
Actores:
Resumen: para establecer una alerta de seguridad en el ambiente domótico controlado, es necesario evaluar ontológicamente el estado de los recursos de los sensores
Precondiciones: el *Proxy* solicita el análisis ontológico
Descripción: Curso Normal: 1. Se solicita el análisis ontológico de la temperatura 2. Se requiere el análisis ontológico de la luminosidad 3. Se requiere el análisis ontológico del movimiento 4. Se determina si existe problema de seguridad
Pos condiciones: se solicita transformar a servicios Web
Observaciones:

Se define todas las actividades tendientes a evaluar el análisis ontológico de la temperatura del dispositivo móvil (tabla 5.10) y el detalle cómo se realiza el análisis ontológico de luminosidad (tabla 5.11).

Tabla 5.10 Caso de uso análisis ontológico temperatura

Caso de Uso: **análisis ontológico temperatura**
Actores:
Resumen: se realiza el análisis ontológico de la temperatura
Precondiciones:
Descripción: Curso Normal: 1. Se recibe el valor en grados Celsius tomado por el sensor 2. Se establece el rango en que se encuentra esta información: 2.1 < 25 : encender calefacción 2.2 25.0 a 25.9 : temperatura ambiente 2.3 > 25.9 : encender ventilación
Pos condiciones:
Observaciones:

Tabla 5.11 Caso de uso análisis ontológico luminosidad

Caso de Uso: **análisis ontológico luminosidad**
Actores:
Resumen: se realiza el análisis ontológico luminosidad
Precondiciones:
Descripción: Curso Normal: 1. Se recibe el valor en lux que es el nivel de luminosidad 2. Se establece el rango en que se encuentra esta información: 2.1 < 700 lx : día 2.2 700 lx a 800 lx : ocaso 2.3 800 lx : noche
Pos condiciones:
Observaciones:

Se explica el detalle de cómo se realiza el análisis ontológico de movimiento (tabla 5.12) y se define todas las actividades relacionadas con la transformación de las acciones del agente a servicios web para indicar la alerta de seguridad y activar la cámara (tabla 5.13).

Tabla 5.12 Caso de uso realizar análisis ontológico movimiento

Caso de Uso: **realizar análisis ontológico movimiento**
Actores:
Resumen: se realiza el análisis ontológico del movimiento
Precondiciones:
Descripción: Curso Normal: 1. Se revisa la información recibida por el sensor en un rango de 6 metros 2. Si establece la alerta en base a: 2.1 Valor booleano verdadero: hay movimiento 2.2 Valor booleano falso: no hay movimiento
Pos condiciones:
Observaciones:

Tabla 5.13 Caso de uso transformar a servicios Web

Caso de Uso: **transformar a servicios Web**
Actores:
Resumen: Se transforman las acciones de los agentes en servicios Web
Precondiciones:
Descripción: Curso Normal: 1. Se obtiene el análisis ontológico de los sensores: temperatura, luminosidad y movimiento 2. Se transforma las acciones del agente a servicios Web 3. Se envía los servicios Web al *Proxy*
Pos condiciones:
Observaciones:

5.1.1. Justificación de los patrones software a utilizar

En el modelo Web interoperable del capítulo anterior se planteó un diseño sustentado en patrones software robustos que apoyaron en establecer mecanismos que permitieron gestionar efectivamente el control de una interrupción de forma interoperable y aplicado a entornos Web, este diseño permitió la organización

estructurada del MVC, se potenció 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 el control de interrupciones en la transmisión de video en tiempo real.

Como muestra la Figura 5.3 se ha establecido un incremento funcional en el subsistema externo que permite el monitoreo constante de los sensores y el análisis ontológico para determinar una alerta de seguridad y activar el servicio de video streaming en tiempo real, en este caso se ha utilizado el *Observer*.

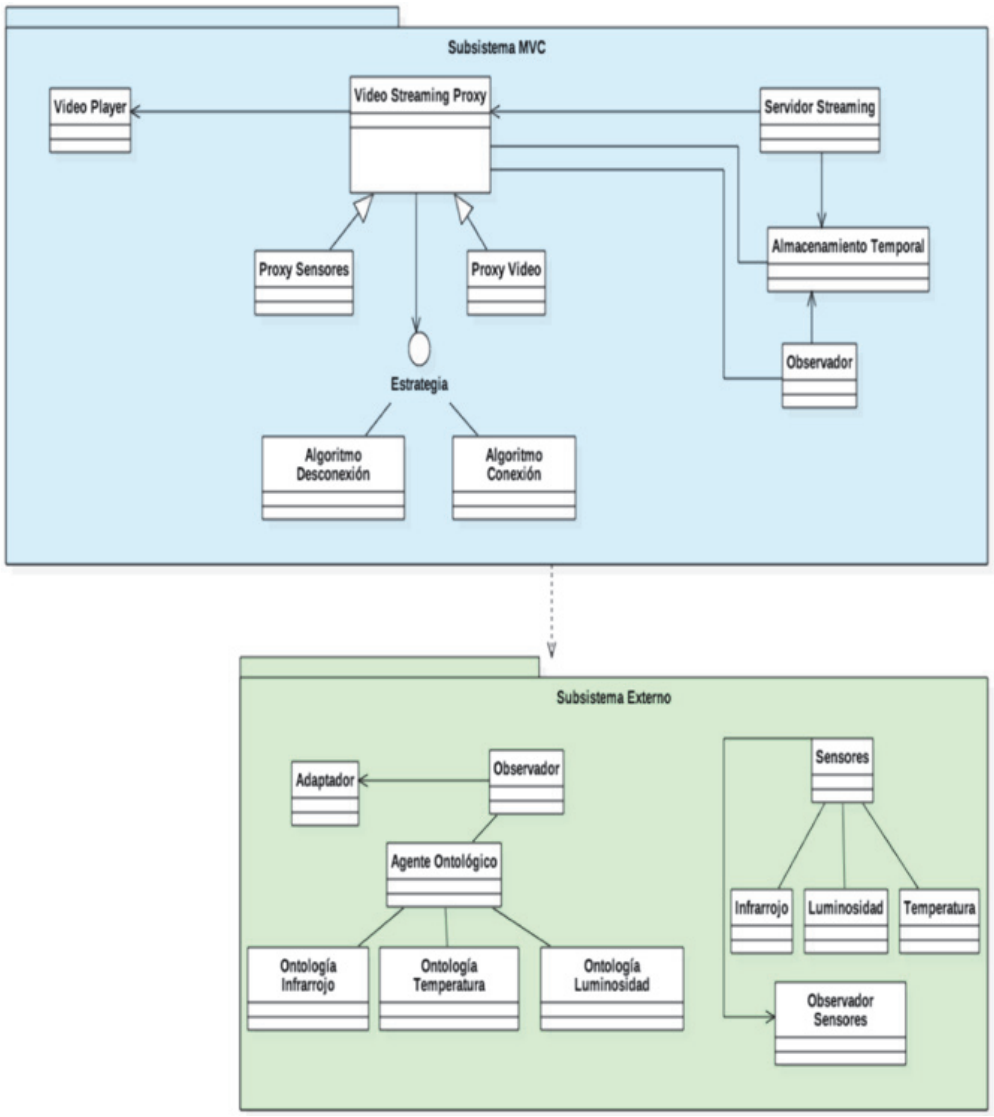

Figura 5.3 Modelo video streaming en tiempo real basado en sensores

Se mantiene la interoperabilidad mediante el *Adapter* que permite la transformación de las acciones del agente a servicios Web.

Se describe a continuación el subsistema MVC:

- *Modelo*: está ubicado en el servidor de aplicaciones, en este se encuentra el almacenamiento temporal donde se almacena la información capturada por la cámara que no puede ser transmitida en tiempo real ya que ha ocurrido una interrupción.
- *Vista*: se despliega en los dispositivos inalámbricos utilizando el *Composite*. Al ocurrir una alerta de seguridad, el cliente recibe un mensaje de emergencia y solicitud de ingreso a la dirección ip establecida. En el video player se despliega entonces toda la información que captura la cámara en tiempo real de lo que sucede en el ambiente controlado que ha sido transgredido. En caso de ocurrir una interrupción, se despliega un mensaje *reconectando…* el cual permanece activo hasta cuando el canal de comunicación se restablece y la visualización del video continúa. Todo lo que ha ocurrido en el tiempo de interrupción es registrado en el repositorio del modelo, estos videos almacenados pueden ser desplegados en la vista en modo *bajo demanda*.
- *Controlador*: se encuentra en el servidor de aplicaciones, para complementar su accionar se ha utilizado los patrones software: *Proxy* y *Strategy*. El patrón *Proxy* cumple la labor de intermediario entre el servidor de video streaming y el reproductor, una de sus actividades principales es solicitar al subsistema externo específicamente al *Observer*, el análisis del estado de los sensores y definir si hay problemas de intrusos en el ambiente controlado, en este caso el *Proxy* se encargara de generar la alerta de seguridad, solicitando se encienda la cámara y el despliegue del video capturado en tiempo real en el dispositivo móvil. Al ocurrir una interrupción en la transmisión del video, el *Proxy* invocará al patrón *Strategy* para que se ejecute el algoritmo de interrupción. Al momento de restablecerse la comunicación, el *Proxy* se encargará de solicitar la ejecución del algoritmo de conexión.

Se describe también el subsistema externo:

- En el lado del modelo se ubica el almacenamiento de la información de los sensores: movimiento, luminosidad y temperatura. Se fortalece con el *Observer* de los sensores que constantemente está monitoreando el estado de los mismos y se lo comunica al *Proxy*.
- En el lado del controlador el *Proxy* envía la información de los sensores al *Observer* del subsistema externo para que realice el análisis ontológico correspondiente y define si ocurre una alerta de seguridad. En caso de ocurrir este evento las acciones del agente ontológico son transformados a servicios Web a través del *Adapter* y enviados al *Proxy*.

Es necesario detallar la funcionalidad del aplicativo de video streaming en tiempo real basado en sensores. La información de los sensores continuamente se actualizan en el subsistema externo, la supervisión del cambio de estado de dichos sensores lo realiza el patrón *Observer* que se encarga de remitir esta información al patrón *Proxy*.

En este caso el *Proxy* solicita el servicio del *Observer* del subsistema externo el cual define que hay una alarma de inseguridad en el ambiente domótico controlado, como resultado del análisis ontológico del estado de los sensores: movimiento, luminosidad y temperatura.

Todo este accionar ontológico es traducido a servicios Web aplicando el *Adapter*. A partir de esta notificación de alarma, el *Proxy* solicita la activación de la cámara y toda la información capturada es transmitida en tiempo real en la interfaz gráfica del dispositivo cliente.

Durante la transmisión en línea puede ocurrir una interrupción, frente a este evento el *Proxy* solicita al patrón *Strategy* la ejecución del algoritmo de interrupción realizándose las siguientes actividades:

- solicitud de almacenamiento de toda la información que captura la cámara en el repositorio del modelo mientras dure la interrupción.

- despliegue en la vista del mensaje que esta desconectado e inicio del bucle de reconexión.

Cuando el bucle de reconexión indica al *Proxy* que se ha superado el problema en el canal, solicita al patrón *Strategy* la ejecución del algoritmo de conexión que permite el despliegue de lo capturado por la cámara en tiempo real y proporcionar al cliente la opción de visualizar el video almacenado en el repositorio en el modo: *bajo demanda*.

5.1.2. Diseño arquitectónico y de despliegue

En el diseño se define la arquitectura de la propuesta que soporte los requisitos y dirija a implantar el modelo de tiempo real con sensores basado en patrones software. El diagrama de clases del sistema se expone en la Figura 5.4.

A continuación se explica la relación entre el modelo de patrones software y las clases que 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: *ServerStreaming*, *Almacenamiento* y *DataListener*.
- Vista y patrón *Composite*: las clases que son ejecutadas para lograr el despliegue del video streaming en el dispositivo cliente son: *VideoPlayer*, *URLStreaming* y *VideoView*.
- Controlador: en este se define el patrón *Proxy* el cual está representado por las clases *Proxy*, *SensorProxy* y *VideoProxy*. El patrón *Strategy* también fortalece la capa de control implementado mediante las clases: *ActionControler*, *InterrupciónAction* y *ReconexiónAction*.

El observador externo que permite evaluar el estado de los sensores está representado por la clase *SensorListener*, controla el almacenamiento temporal de las clases que representan el estado de los sensores *Temperatura*, *Luminosidad* y *Movimiento*, las cuales son observadas para definir un cambio de estado, para solicitar posteriormente el análisis ontológico.

Se ha adicionado un observador *OntologiaListener* que evalúa el análisis ontológico de los recursos compuesto de las clases *AgenteOntologico*, *OntologiaTemperatura*, *OntologiaLuminosidad* y *OntologiaMovimiento*; estas clases trabajan en conjunto para determinar proactivamente si hay un intruso en el ambiente controlado, las acciones de los agentes son transformadas a servicios Web mediante *AdaptadorServiciosWeb*.

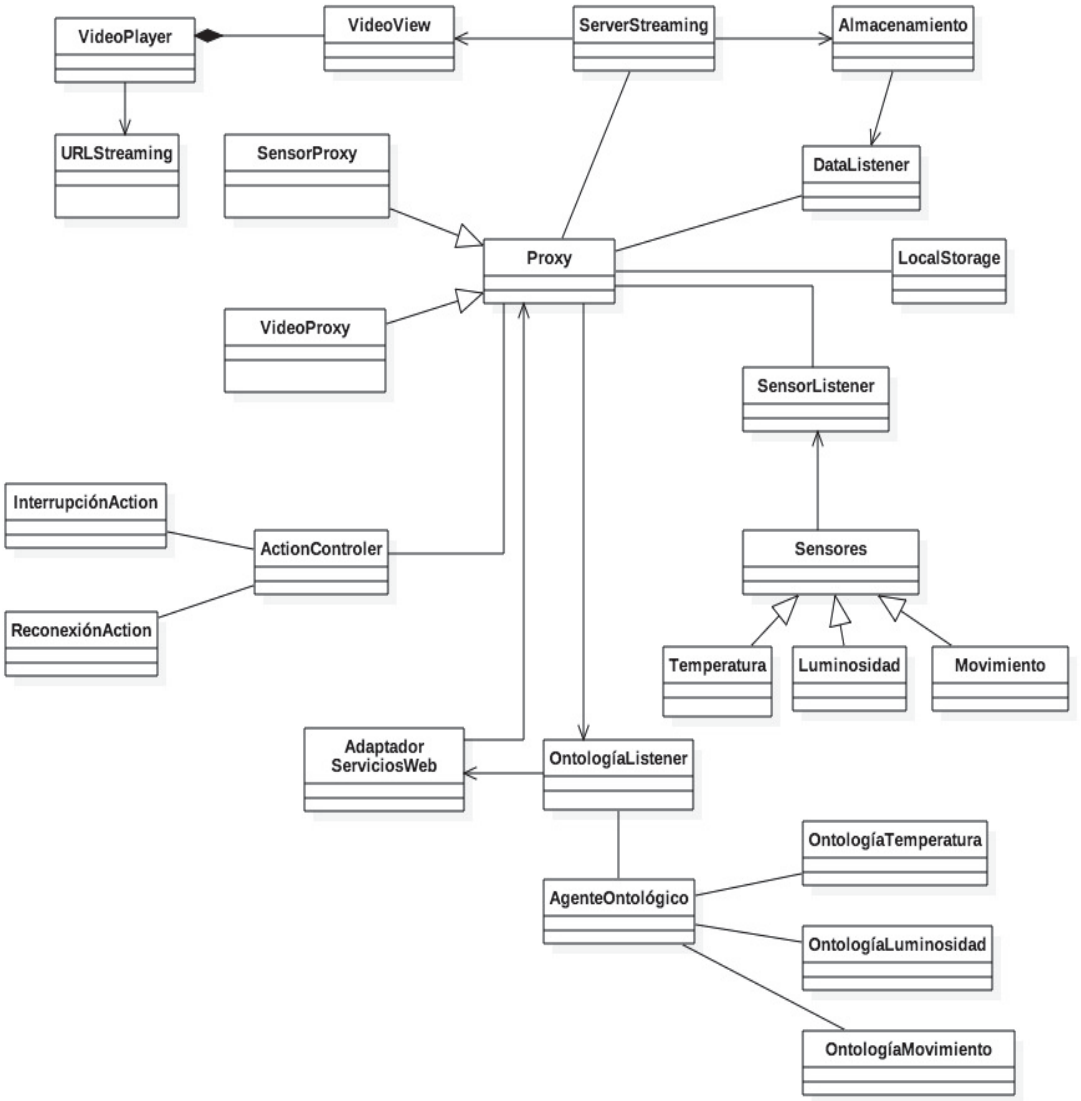

Figura 5.4 Diagrama de clases

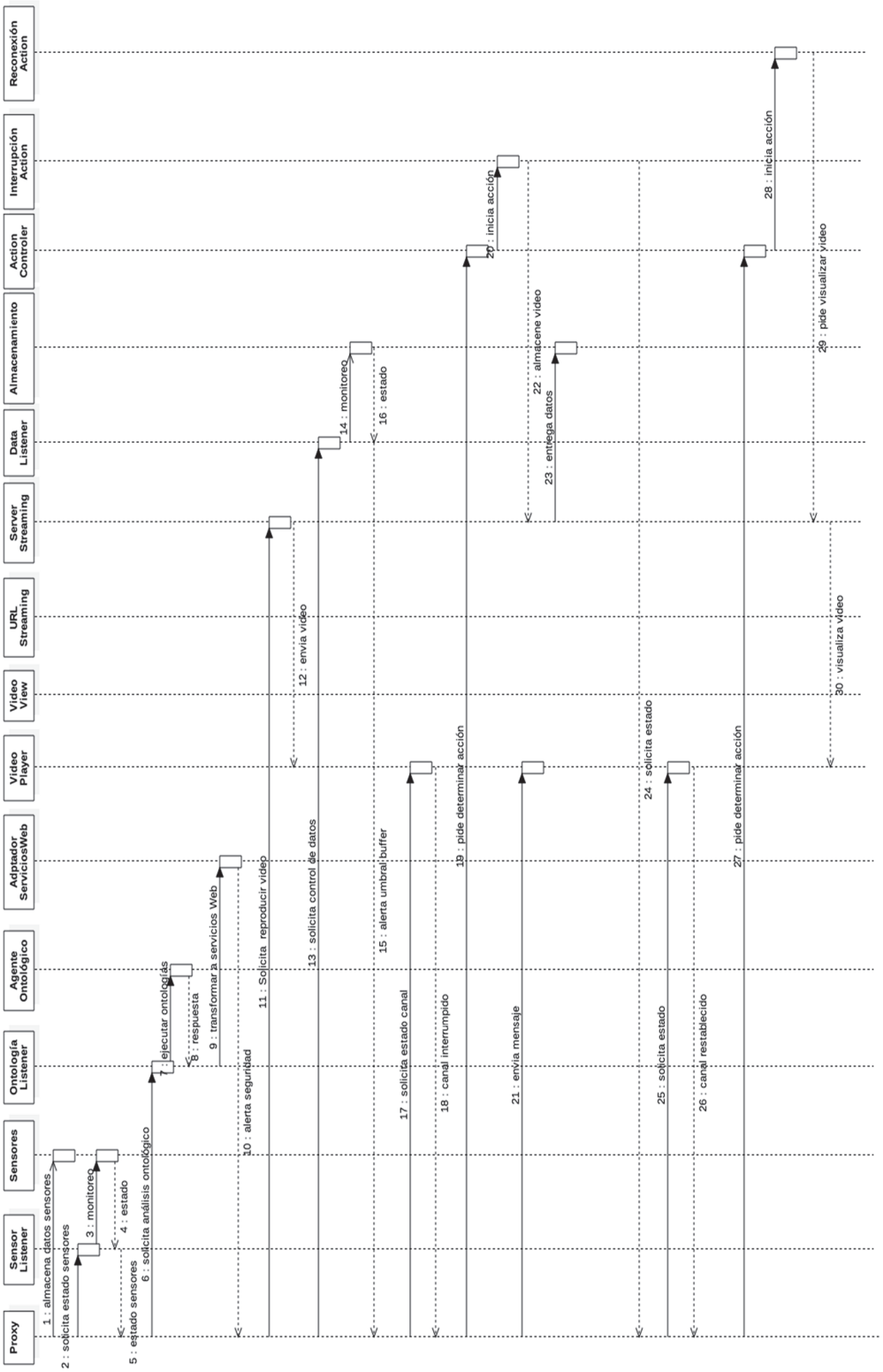

Figura 5.5 Diagrama de secuencia

En la Figura 5.5 se muestra el diagrama de secuencia, comienza con el observador del estado de los recursos de los sensores y el análisis ontológico correspondiente para determinar una alerta de seguridad. En caso de ocurrir este evento se inicia el proceso de comunicación del video streaming en tiempo real entre las clases definidas anteriormente para solventar la solicitud y despliegue del servicio de video streaming, una vez establecido este enlace es necesario activar la escucha del control de datos. Si el almacenamiento presenta nivel superior al umbral, este evento es notificado a la capa controladora. Para asegurar que es una interrupción se analiza el canal de comunicación mediante el observador quien notifica de este evento al *Proxy*. Este a su vez se encargara de solicitar se guarden los flujos capturados por la cámara en un almacenamiento temporal. Una vez que se indique que el canal se ha restablecido, otras acciones son ejecutadas tendientes a solicitar al servidor de video streaming continúe con el envío de los flujos.

En la Figura 5.6 se puede observar el diagrama de despliegue, el *Servidor de Aplicación* contiene a las clases que permiten el almacenamiento temporal del estado de los sensores *Temperatura*, *Luminosidad* y *Movimiento*. También se alojan las clases encargadas del análisis ontológico de los mismos con la finalidad de determinar si se ha dado una alerta de seguridad, el observador *AgenteOntologico* solicita el servicio a *OntologiaTemperatura*, *OntologiaLuminosidad* y *OntologiaMovimiento*; si este evento ocurre los servicios del agente ontológico son traducidos a servicios Web a través de la clases adaptadora *AdaptadorServiciosWeb*; esta novedad es recibida por el *Proxy* quien realiza el pedido a *ServerStreaming* para que inicie el servicio de video streaming en tiempo real.

El dispositivo móvil aloja al *VideoPlayer*, *URL* y *VideoView* que son las clases que interactúan en el despliegue del video. Al iniciar esta visualización el *Proxy* solicita el trabajo de observador a *DataListener* quien continuamente está revisando al *Buffer* y define alerta de umbral superado. 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 lo que está capturando la cámara mientras dure la interrupción y pide se evalúe el canal y determine su restablecimiento.

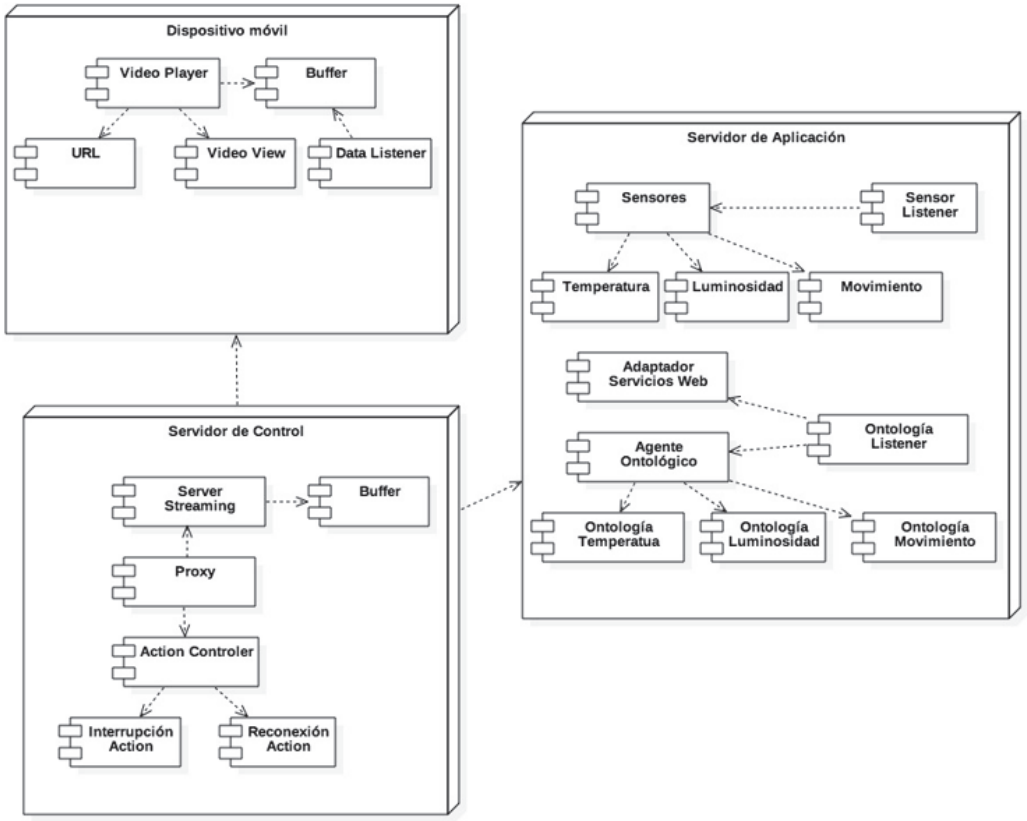

Figura 5.6 Diagrama de despliegue

Al restablecerse el canal el *Proxy* recibe esta notificación y pide al *ActionControle*r la ejecución del algoritmo correspondiente, para esta situación sería *ReconexiónAction* mediante el cual el *Proxy* solicita al *ServerStreaming* continúe con la transmisión en línea del video.

5.2. Modelo matemático de rendimiento

Como se ha mencionado la solución de video en tiempo real se basa en la solución Web interoperable, realizándose incrementos de funcionalidades específicamente en el uso de sensores que activan la cámara y el inicio del video streaming en tiempo real si se da una alerta de seguridad. El tiempo que toma la comunicación de los sensores con la cámara y activar la transmisión del video es menor, podría compararse con un *Tstart* que se encuentra en el rango de 2 a 4 s; debido a que nuestro interés es la transmisión en tiempo real del video streaming, este tiempo no ha sido considerado en el análisis del modelo matemático.

Se mantiene la interoperabilidad a través de agentes ontológicos traducidos a servicios Web. El observador revisa el estado de sensores y mediante el análisis ontológico define una alerta, en base a la cual inicia la transmisión del video en tiempo real.

Al transmitir al dispositivo cliente lo que está capturando la cámara en tiempo real puede darse una interrupción, esta es definida a través del observador del canal que lo notifica al *Proxy* el cual gestiona el almacenamiento del video que no está siendo reproducido en el cliente en un buffer temporal, el cual podrá ser visto una vez que el cliente solicite la visualización de este video bajo demanda. Al ocurrir el restablecimiento del canal se realiza la reproducción inmediata de lo capturado por el servidor de video streaming (cámara) en el cliente.

Basado en lo expuesto el modelo matemático aplicado para el control de interrupciones definido en el capítulo 3, es utilizado en esta propuesta.

$$T_{exec} = T_{start} \ + \sum_{i=1}^{n} \left(T_{data_i}\right) + \sum_{i=1}^{n} T_{int_i} + T_{data_n}$$

Donde:

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

$\sum_{i=1}^{n} \left(T_{data_i}\right)$ es el tiempo total de *Tdata* transmitidos.

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

T_{data_n} es el último frame transmitido.

El evento de intrusos en el ambiente controlado genera la transmisión del video en tiempo real, se ha analizado el tiempo máximo en que pueden robar y se ha definido como máximo 10 minutos, por lo que la duración del video a analizar es de 600 s.

El tiempo de interrupción es importante en este servicio de video streaming por lo que se ha establecido para el análisis dos escenarios: el mejor escenario que tiene

interrupciones de menor duración, de 10 a 15 segundos y el peor escenario presenta interrupciones de mayor duración, de 70 a 80 segundos.

Se han realizado 5 pruebas reales con 3 interrupciones en el mejor escenario, puede observarse en la Figura 5.7, el tiempo total de ejecución es afectado en menor grado. En la Figura 5.8 se detallan las pruebas reales efectuadas en el peor escenario donde el tiempo de ejecución es totalmente afectado por las interrupciones. En la Figura 5.9 puede observarse la influencia del tiempo de interrupción en el tiempo total de ejecución del video.

	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b	c	Texec
									∑(Tdatai)	∑(Tinti)	Tstart + b + c + Tdatan
1	4,00	147,45	12,29	157,61	11,63	123,67	11,78	171,27	428,73	35,70	639,70
2	2,80	127,79	10,96	119,02	10,55	165,87	13,84	187,32	412,68	35,35	638,15
3	2,66	130,88	14,99	122,98	12,23	142,75	14,54	203,39	396,61	41,76	644,42
4	3,06	123,48	11,33	133,56	12,14	132,54	12,38	210,42	389,58	35,85	638,91
5	2,83	152,16	12,69	161,99	13,89	136,87	11,38	148,98	451,02	37,96	640,79

Figura 5.7 Mejor escenario

	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b	c	Texec
									∑(Tdatai)	∑(Tinti)	Tstart + b + c + Tdatan
1	4,00	147,45	73,74	157,61	69,78	123,67	70,68	171,27	428,73	214,2	818,20
2	2,80	127,79	65,76	119,02	63,30	165,87	83,04	187,32	412,68	212,1	814,90
3	2,66	130,88	89,94	122,98	73,38	142,75	87,24	203,39	396,61	250,56	853,22
4	3,06	123,48	67,98	133,56	72,84	132,54	74,28	210,42	389,58	215,1	818,16
5	2,83	152,16	76,14	161,99	83,34	136,87	68,28	148,98	451,02	227,76	830,59

Figura 5.8 Peor escenario

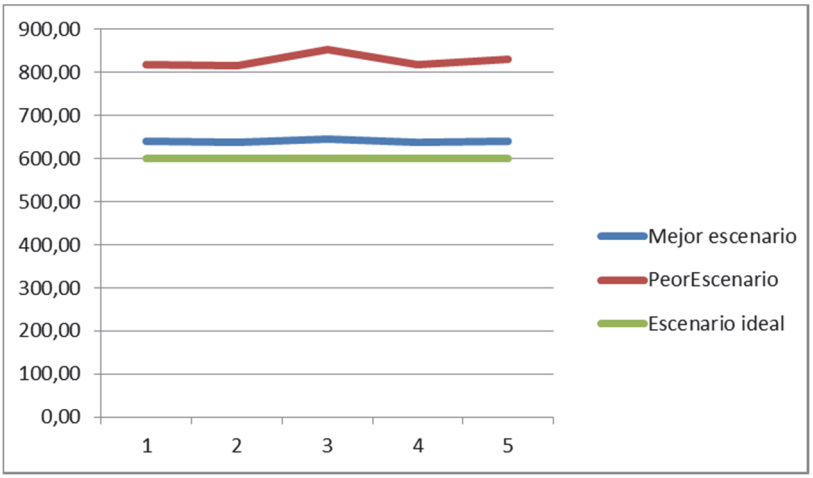

Figura 5.9 Análisis de escenarios

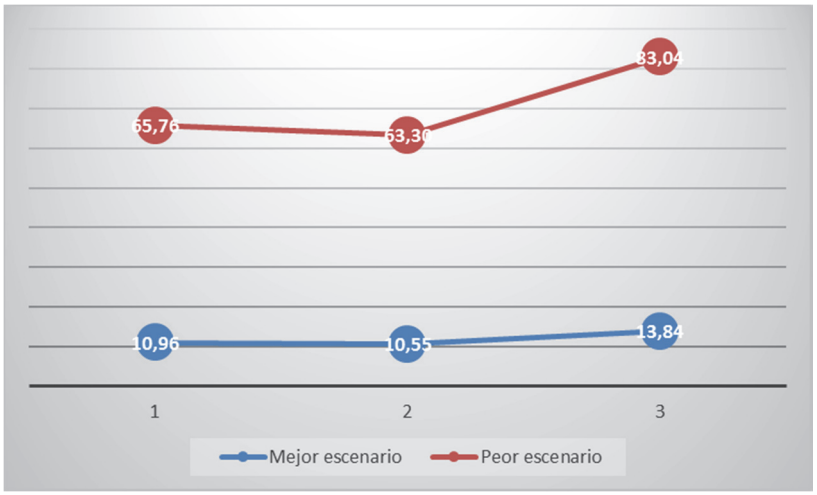

Figura 5.10 Análisis de interrupciones por escenario

Es importante definir el mayor tiempo de duración de una interrupción, de acuerdo a la Figura 5.10 seria 83,04.

Para la ejecución de este modelo ha sido necesario utilizar un almacenamiento temporal donde se alojan los videos capturados por la cámara mientras dure una interrupción. Se ha establecido como máximo el tamaño de video almacenado de 100 s. Este tamaño se ha establecido en base al valor máximo de una interrupción definido en la Figura 5.10.

Para evaluar la cantidad de memoria utilizada, se define como velocidad del canal de 10Mbps en ADSL entonces se determina el número de bits que transmite con la siguiente ecuación:

$$Número\ de\ bits = velocidad\ del\ canal * tiempo\ de\ la\ interrupción$$

$$Número\ de\ bits = 100 * 10 = 1000\ \ bits$$

Para obtener bytes lo dividimos para 8, obteniendo 1000/8 = 125 MB.

El uso de 125 MB de almacenamiento es un costo razonable a pagar al recibir los beneficios proporcionados por un mecanismo que controla interrupciones. Además, tal cantidad de memoria no afecta a los grandes almacenamientos locales o en la nube que existen en la actualidad.

5.3. Análisis de la implantación experimental

Se ha construido la arquitectura denominada *Interrupt Control Architecture of Real Time Video streaming* (*ICARTS*) la cual gestiona las interrupciones del video streaming en tiempo real a través del uso de agentes Web. Esta arquitectura está basada en la red de sensores que son censados constantemente, se solicita los servicios de los agentes Web interoperables que permiten evaluar de manera ontológica el ambiente domótico y establecer alertas en caso de que un evento suceda. Al darse una alerta, se activa la cámara y el envío del video streaming en tiempo real al cliente. En esta transmisión en línea puede ocurrir una interrupción inalámbrica, la arquitectura ICARTS, reduce el impacto negativo que ocasiona una interrupción de video streaming en el usuario, ya que proporciona el servicio de reconexión automático y la grabación del video streaming bajo demanda de lo que sucedió durante la interrupción, para que pueda ser visualizado cuando el cliente lo requiera.

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

A continuación se explica la relación entre el modelo de patrones software y las clases que han sido definidas en el diagrama de clases, Figura 5.11:

- Modelo y patrón *Observer*: las clases que permiten el almacenamiento temporal y el observador que analiza constantemente el estado del buffer son: *CamaraService*, *Live*, *Storage* y *DataListener*.
- Vista y patrón *Composite*: las clases que son ejecutadas para lograr el despliegue del video streaming en el dispositivo cliente son: *VideoPlayer*, *URL* y *VideoView*.
- Controlador: define el patrón *Proxy* el cual está representado por la clase *Manage*. El patrón *Strategy* también fortalece la capa de control implementado mediante las clases: *SensorControl*, *InterruptionAction* y *ConnectionAction*.

El observador externo que permite evaluar el estado de los sensores está representado por la clase *DataListener*, controla el almacenamiento temporal de las clases

Temperatura, *Luminosidad* y *Movimiento*. También se encuentra alojado el observador *AgentClass* que se encarga de revisar el estado ontológico de los sensores *Temperature*, *Motion* y *Light* a través de *SensorAgent*, *IOntology*, *Vocabulary*, *Alarm* y traducirlas a servicios Web mediante *WebServiceControl*. Las clases encargadas de capturar la información a través de la cámara y enviarlas al *Proxy* son *CamaraService*, *Config*, *BdConn*, *CamaraCmd* y *CdmControl*.

En la Figura 5.12 se muestra el diagrama de secuencia, inicia con el observador del estado de los recursos de los sensores y el análisis ontológico correspondiente para determinar una alerta de seguridad. En caso de ocurrir este evento se inicia el proceso de comunicación del video streaming en tiempo real entre las clases definidas anteriormente para solventar la solicitud y despliegue del servicio de video streaming, una vez establecido este enlace es necesario activar la escucha del control de datos. Si el almacenamiento presenta nivel superior al umbral, este evento es notificado a la capa controladora.

Para asegurar que es una interrupción se analiza el canal de comunicación mediante el *Observer* quien notifica de este evento al *Proxy*. Este a su vez se encargara de solicitar se guarden los flujos capturados por la cámara en un almacenamiento temporal. Una vez que se indique que el canal se ha restablecido, se activara automáticamente la reproducción de lo capturado por la cámara y el usuario tendrá la opción de visualizar los videos guardados en el almacenamiento temporal en modo *bajo demanda*.

En la Figura 5.13 se puede observar el diagrama de despliegue, el *Servidor de Aplicación* contiene a las clases que permiten el almacenamiento temporal del estado de los sensores para que sean evaluados por el observador. Otro observador solicita el análisis ontológico de los mismos con la finalidad de determinar si se ha dado una alerta de seguridad. Si este evento ocurrió se inicia la solicitud a las clases contenidas en el *Servidor de Video* para que inicie el servicio de video streaming en tiempo real. El dispositivo móvil aloja al *VideoPlayer*, *URL* y *Video View* que son las clases que interactúan en el despliegue del video.

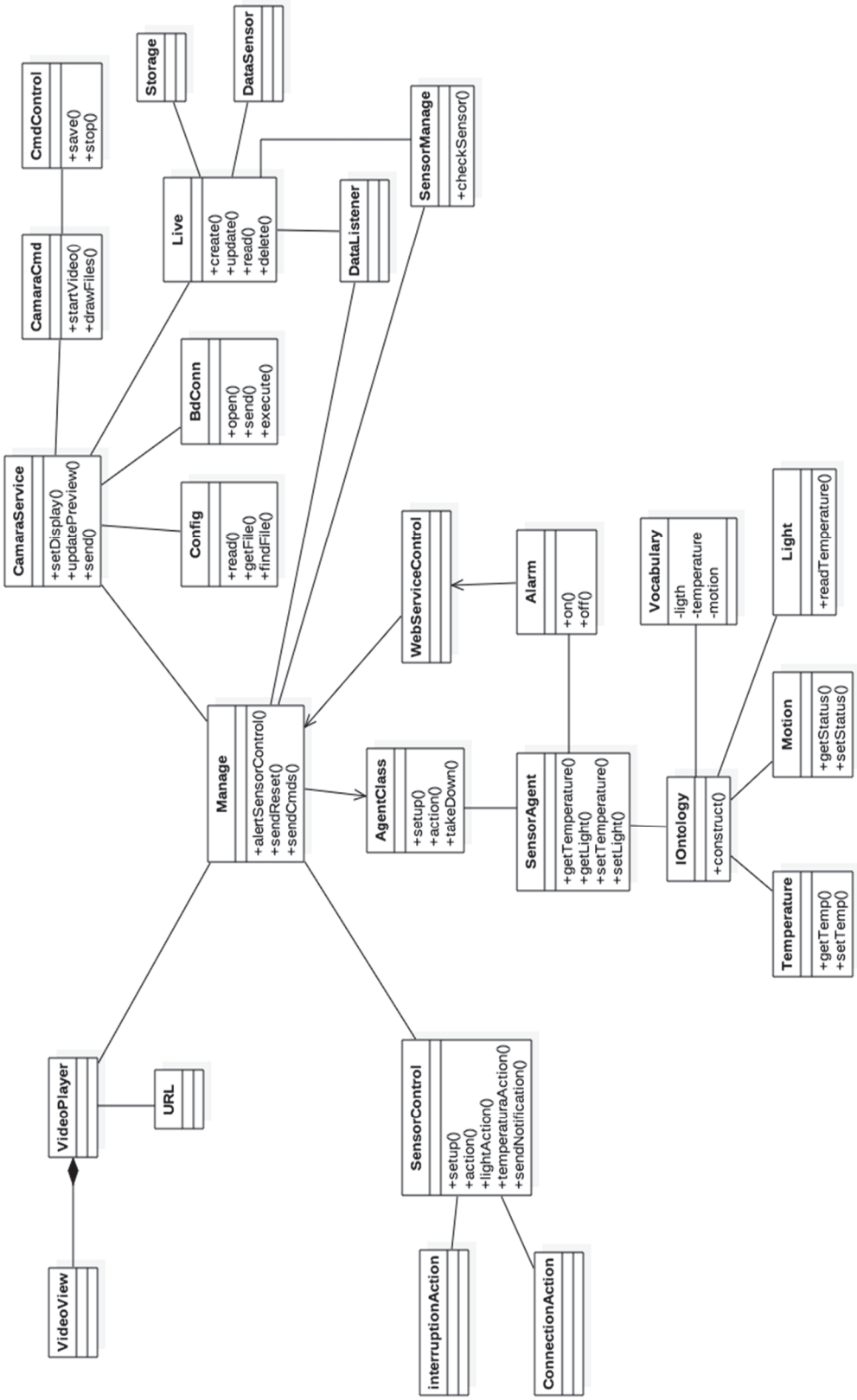

Figura 5.11 Modelo de clases

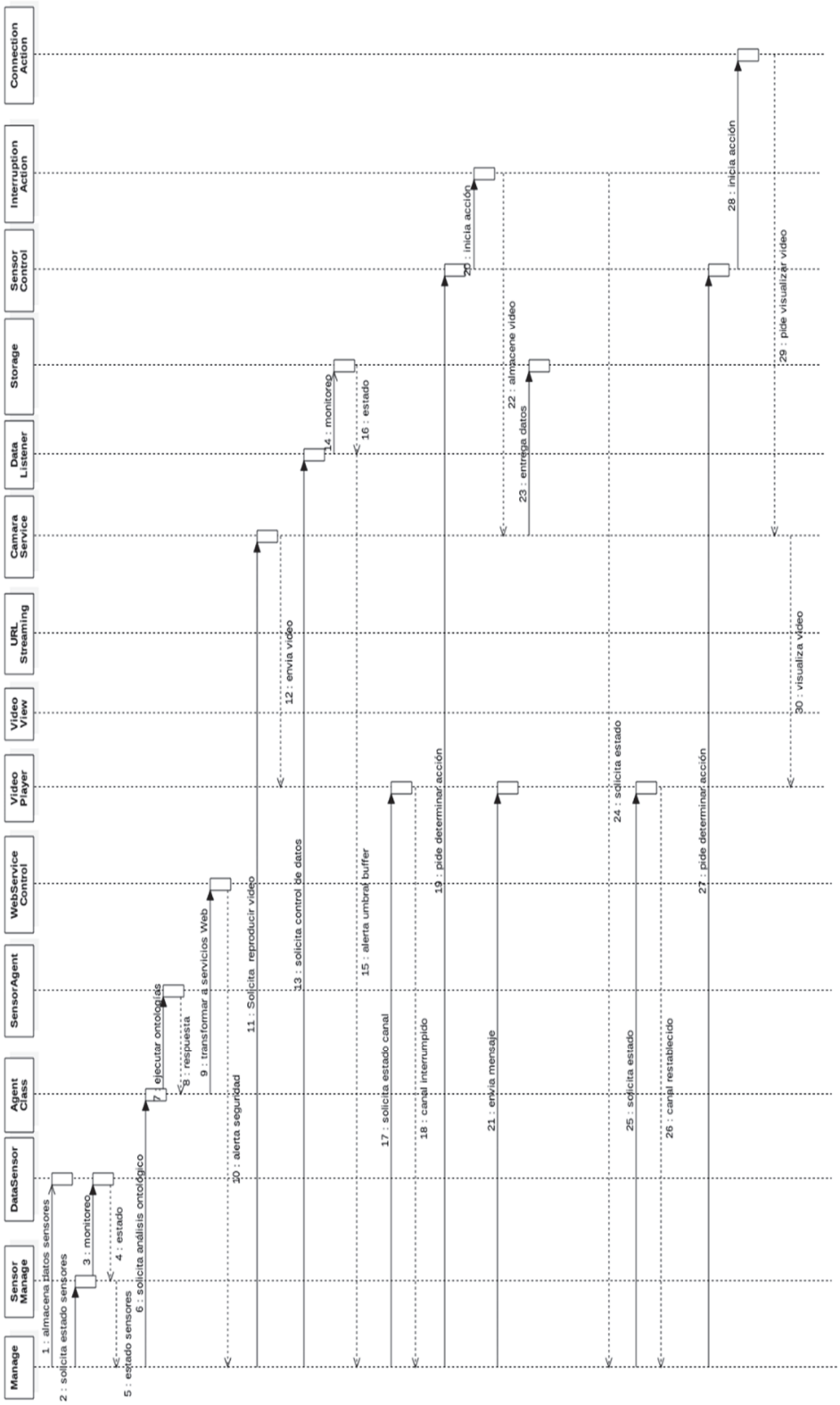

Figura 5.12 Diagrama de secuencia

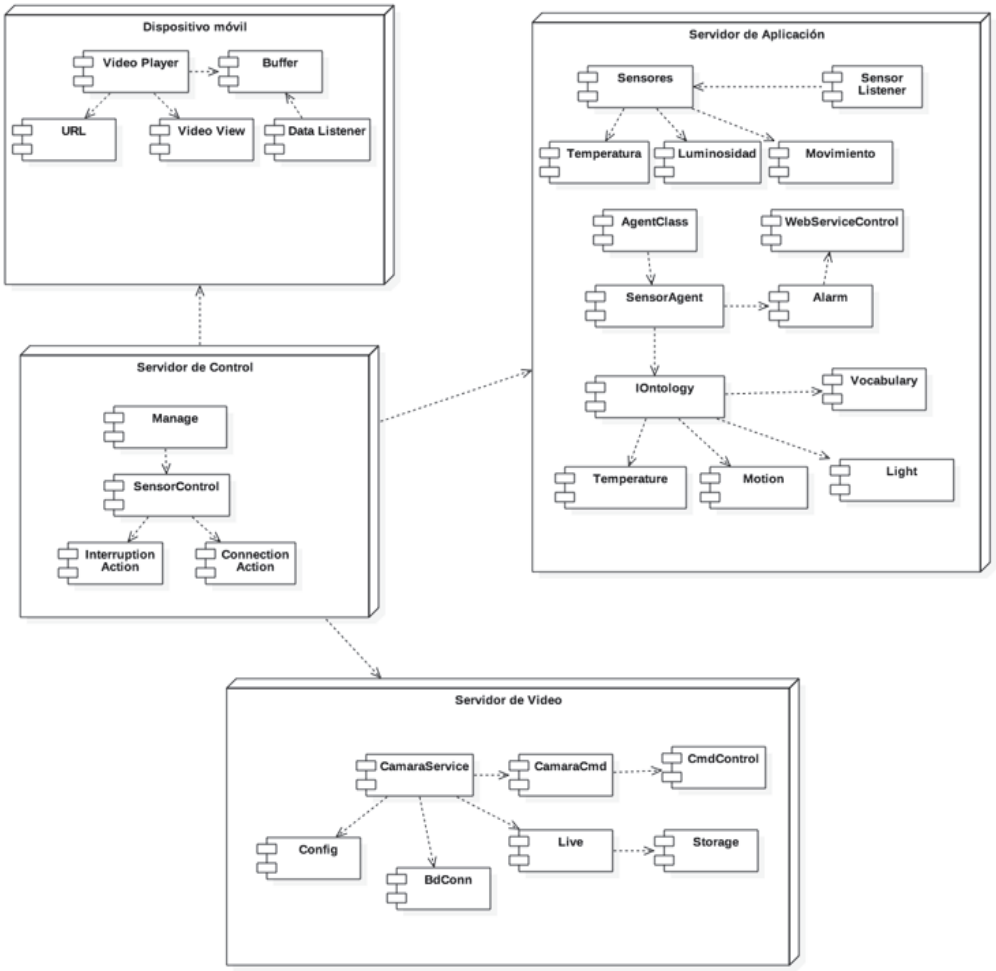

Figura 5.13 Diagrama de despliegue

Al iniciar esta visualización el *Proxy* solicita el trabajo de observador a *DataListener* quien continuamente está revisando al *Storage* y define alerta de umbral superado. Si esta respuesta asegura que hay una interrupción el *Proxy* invoca al servicio del *SensorControl* para que seleccione el algoritmo a ejecutar, en este caso *InterruptionAction*. Este algoritmo se encarga de que el *Proxy* solicite a *CamaraService* servidor de video streaming el almacenamiento temporal de lo que está capturando la cámara mientras dure la interrupción y pide se evalúe el canal y determine su restablecimiento. Al restablecerse el canal el *Proxy* recibe esta notificación y pide al *SensorControl* la ejecución del algoritmo correspondiente, para esta situación será *ConnectionAction* mediante el cual el *Proxy* solicita a *CamaraService* continúe con la transmisión en línea del video.

5.3.2. Desarrollo del modelo con software libre

En el desarrollo de la arquitectura ICARTS (Figura 5.14), como medio de interacción y comunicación se utilizó el servidor de aplicaciones Apache Tompcat, donde se levantan los servicios Web del *SensorAgent* por medio de JADE WSIG. JADE únicamente se encarga del diagnóstico de eventos en los sensores, utilizando *SensorAgent*, que posee tres comportamientos: temperatura, luminosidad, y movimiento.

Para controlar el ambiente, es necesario conocer constantemente el estado del ambiente de prueba, el *Sensor Server* se encarga de la recepción de la información proveniente de los sensores, el *Sensor Proxy* es el encargado de receptar la información del *Sensor Server* y enviarlo a JADE, allí se procesan los datos y la respuesta obtenida regresa al *Proxy*, para posteriormente tomar acciones (encender luces, encender calefacción apagar ventilación...) dentro del ambiente controlado según el análisis ontológico realizado, para lo cual la plataforma presenta dos modos de utilización; el modo normal donde los usuarios se encuentran dentro del ambiente controlado y el modo seguro que arma las alarmas y demás aspectos para preservar la seguridad.

En caso de generarse una alerta el sensor *Proxy* lanza una señal que notifica al *Servidor de Streaming* que active la PiCam realizando las siguientes acciones:

- Notificación al cliente, mediante un mensaje vía SMTP de forma pasiva, para que se conecte a la red revisando el video en tiempo real.
- Activar la secuencia de grabación en el instante que se dispararon las alarmas.
- En caso de conectarse el cliente se despliega el video streaming en tiempo real o permite visualizar el historial de grabaciones.

Para proporcionar video streaming en tiempo real en HTML5, se utilizó como servidor de video streaming a Rpi_cam de RaspberyPi, el cual, envía los frames al *Proxy* video.

El *Sensor Proxy* y el *Video Proxy* depositan su información en un *Proxy* general que se comunica directamente con el cliente permitiéndole interactuar con el ambiente controlado o revisar los videos de seguridad.

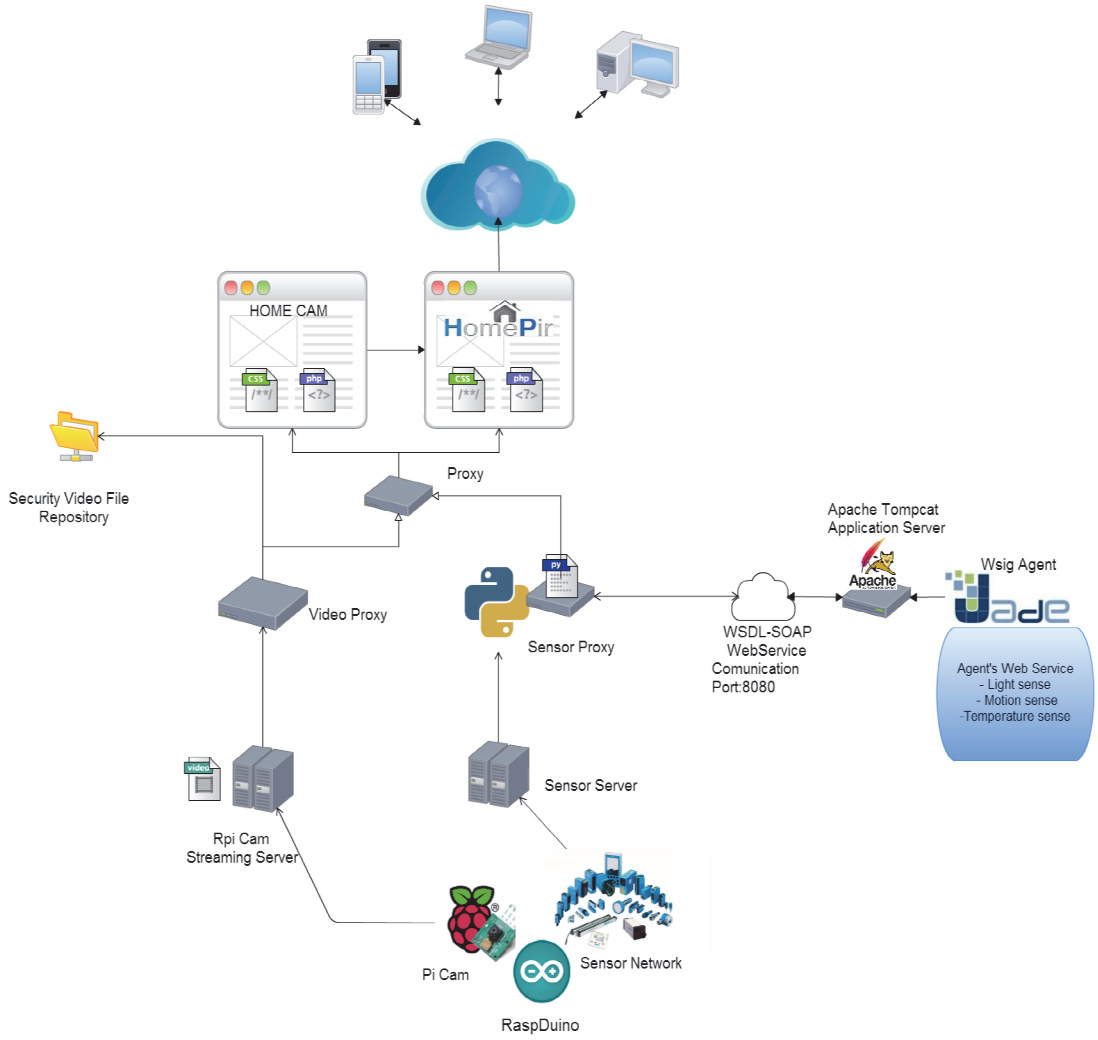

Figura 5.14 Arquitectura ICARTS

Cuando el cliente deja de recibir frames y el servicio se detiene, el *Proxy* envía una señal al *Video Proxy* para que entre en un bucle de reconexión, para que al momento de regresar la cobertura, el servicio de video streaming en tiempo real se reconecte de manera automática. Mientras dure la interrupción, el *Video Proxy* graba el video y lo almacena en un directorio propio de Rpi_Cam; cuando se reanuda la comunicación el *Proxy* enlaza a los dos proxies hijos recuperando el video en tiempo real y permitiendo la interacción con el ambiente controlado.

El Proxy de Control es el encargado de verificar que el estado del canal de comunicación este operativo, a este estado se lo conoce como “conexión”; cuando el *Proxy* detecta que esta condición no se cumple entra en un estado de “interrupción” donde en el cliente aparece un mensaje *RECONECTANDO…* y en el servidor se activa la

petición del servicio de grabación. Cuando el cliente regresa la red se conecta directamente con el video en tiempo real.

En el ambiente JADE se creó el agente *SensorAgent*, el cual presenta tres comportamientos: temperatura, luminosidad y movimiento. De éstos sensores se almacena la información de: identificador, timestamp, medida y unidad. Una instancia de esta ontología en el caso del sensor de temperatura sería:

{ "SensorTemperatura": {

"identificador":"ST-TA3231-1",

"timestamp":{"$date": "2014-01-27T11:14:00Z"},

"medida":25.1, "unidad":"C",

} }

Los comportamientos que ofrece SensorAgent son:

- Temperatura: recibe el valor en grados Celsius tomado por el sensor, donde basados en el comportamiento de temperatura ambiente aceptable, definiendo los siguientes intervalos:
  - < 25 º C : encender calefacción
  - 25.0 ºC – 25.9 ºC : temperatura ambiente, comportamiento sin acción
  - mayor de 25.9 ºC : encender ventilación
- Luminosidad: recibe el valor en lux (nivel de iluminación o densidad luminosa) tomado por la fotorresistencia (sensor de luminosidad) donde basados en el comportamiento de luminosidad se define los siguientes intervalos:
  - < 700 lx: día
  - 700 lx – 800 lx : ocaso
  - 800 lx : noche
- Movimiento: recibe un valor booleano verdadero al detectar movimiento o un valor booleano falso al no detectarlo; en un rango de 6 metros.

La funcionalidad de Rpi Cam se centra en el proceso de raspimjpeg que accede a los datos de la cámara. La Figura 5.15 muestra los componentes principales. Las líneas azules indican los flujos de datos. Las líneas rojas indican el control. Se realiza una conexión con la cámara y se genera flujos jpeg, que son capturados en el directorio directorio / dev / shm / MJPEG, ésta carpeta es una memoria RAM. El vídeo se almacena en el formato H.264.

Esta información puede ser visualizada a través de una dirección URL en un navegador, entonces la página principal (index.php) utiliza cam_new_pic.php. Raspimjpeg se puede controlar mediante el envío de comandos en una pila llamada FIFO en la carpeta /var /www. Los comandos dan acceso a la mayoría de los ajustes de la cámara y también la posibilidad de iniciar o detener los procesos de captura.

El diseño Web adaptable fue aplicado en el desarrollo del aplicativo, conocido como Responsive Web Design, son prácticas que se aplican al diseño Web, orientados a la experiencia del usuario cuando accede a una aplicación Web para permitir su acceso desde cualquier medio como: móviles, tablets, pantallas, sin importar la resolución en el computador.

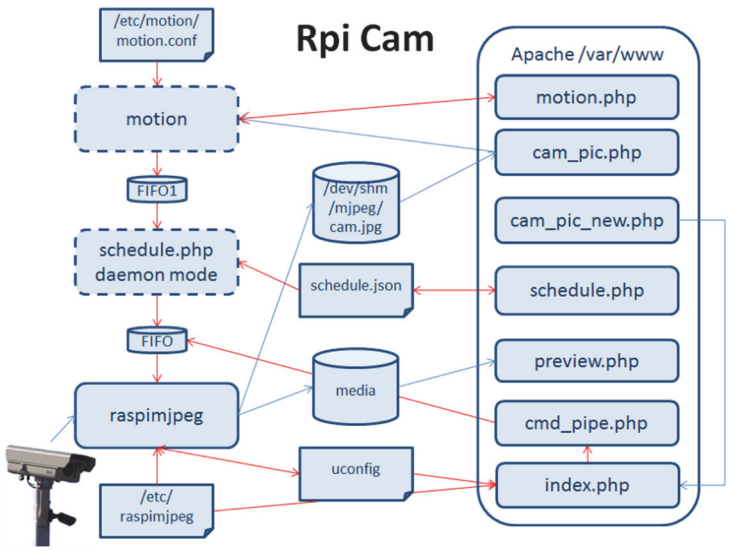

Figura 5.15 Funcionalidad de Rpi Cam

Para aplicar Web responsive en la arquitectura ICARTS, se utilizó *Cascading Style Sheets* (*CCS3*) [23], mecanismo para adicionar estilo y formato a los documentos en los sitios Web.

5.3.3. Resultados experimentales

Se trabajó con el servidor de video streaming interno de la Rasberry PI, para la transmisión de video utiliza el estándar H.264 [160] que define un CODEC de alta compresión y regula la calidad de la imagen con tasas binarias notablemente inferiores al estándar previo (H.263). Además Raspberry PI no tiene una tarjeta de audio para el procesamiento de esta información.

El escenario de pruebas se ha conformado por los siguientes equipos, Figura 5.16:

- Raspberry Pi: se ha utilizado Raspberry Pi B+, procesador: 900MHz de cuatro núcleos de CPU ARM Cortex-A7, 500 GB de RAM, 4 puertos USB, 40 pines GPIO, interfaz de la cámara (CSI), interfaz de pantalla (DSI), ranura para tarjeta Micro SD y núcleo de gráficos VideoCore IV 3D.

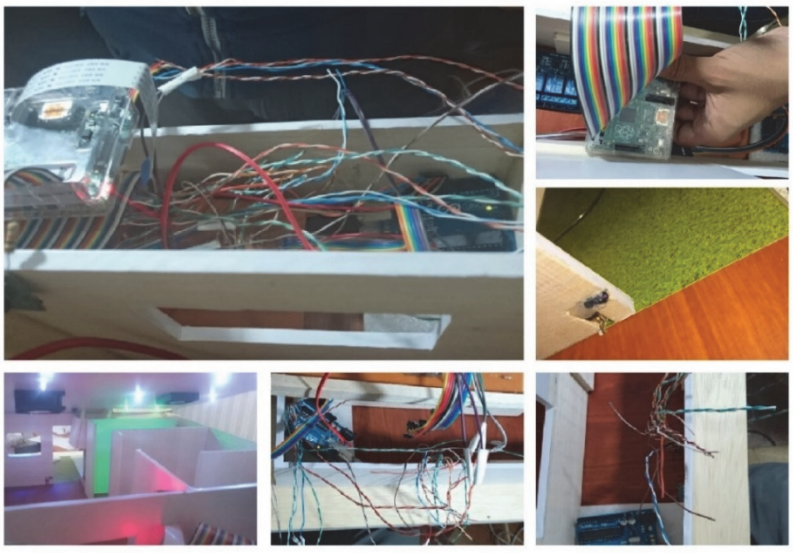

Figura 5.16 Escenario experimental

- Arduino: se han usado *Arduino Uno*: microcontrolador Atmega328, voltaje de operación 5V, voltaje de entrada (Recomendado) 7 – 12V, voltaje de entrada (Límite) 6 – 20V, pines para entrada- salida digital. 14 (6 pueden usarse como salida de PWM), pines de entrada analógica, 6 Corriente continua por pin IO 40 mA, corriente continua en el pin 3.3V 50 mA, memoria Flash 32 KB (0,5 KB ocupados por el bootloader) SRAM 2 KB EEPROM 1 KB Frecuencia de reloj 16 MHz.
- Camara 20.7 MP, flash LED, sensor 1/2.3'', geo-tagging, foco táctil, detección de rostro y sonrisa, fotos panorámicas 3D, estabilizador de imagen, HDR, sensor Exmor RS, video 2160p@30fps, autofoco continuo, luz de video, cámara frontal 2.2MP 1080p.

Se ha construido un ambiente domótico controlado para evaluar el funcionamiento del video streaming en tiempo real, como puede observarse en las Figuras 5.17, 5.18 y 5.19, se han ubicado 2 sensores de luz parte delantera y trasera de la vivienda, 2 sensores de temperatura sala y cocina, 4 sensores infrarrojos para detección de movimiento (entrada delantera, trasera, cocina, baño) y 2 actuadores para enfriamiento (ventiladores).

Los sensores utilizados:

- Sensor de luz: dimensiones 65x11x13 mm, serie Fotoresistor-BH1750, mediciones 1-65535 LX, muestreo 2s.
- Ds28b20 temperatura digital: cada dispositivo tiene un código de serie 64-bit único almacenado en una ROM a bordo, capacidad multipunto simplifica las aplicaciones de detección de temperatura distribuidos, puede ser alimentado desde la línea de datos. Rango de suministro de energía es3.0V a 5.5V. Mide temperaturas de -55 ° C a + 125 ° C (-67 ° F a + 257 ° F) ± 0,5 ° C Precisión de -10 ° C a + 85 ° C. Resolución termómetro es seleccionable por el usuario de 9 a 12 bits de y convierte la temperatura a la palabra digital de 12 bits en 750 ms (máx.)
- Barrera Infrarroja: tipo de detector: fototransistor, dimensiones (L x W x H en mm): 10.2 x 5.8 x 7, distancia de operación pico: 2.5 mm, intervalo de

operación para una variación de corriente de colector mayor al 20 %: 0.2mm a 15 mm, corriente de salida típica bajo prueba: Ic = 1 mA, filtro de bloqueo de luz ambiental, longitud de onda del emisor: 950 nm.

Figura 5.17 Ambiente domótico parte delantera

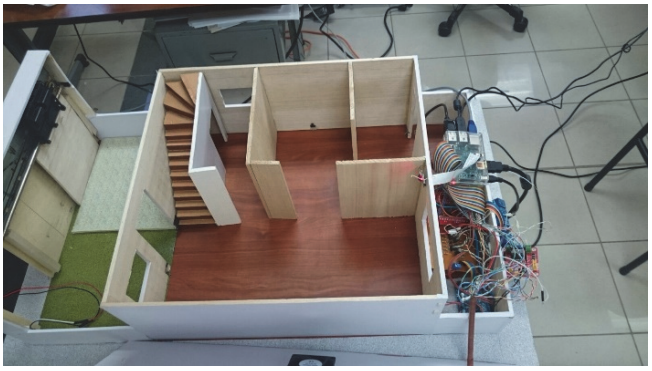

Figura 5.18 Ambiente domótico interiores

Figura 5.19 Ambiente domótico sensores

En la arquitectura planteada pudo evaluarse el tiempo de activación de los sensores, tiempo de llegada de la notificación al cliente, debido a que nos interesa evaluar la transmisión de video streaming en tiempo real nos hemos enfocado en evaluar los tiempos de transmisión e interrupción.

El primer escenario donde se hicieron las pruebas reales es en Quito a varios kilómetros de distancia de la universidad donde se encuentra alojado el ambiente domótico construido, en dichas pruebas se estableció como tiempo de interrupción 80 segundos, los datos reales se muestran en la Figura 5.20 en la que indica el tiempo total de ejecución y el de interrupción.

En base a la Figura 5.21, se analiza que el tiempo de ejecución está compuesto del 50% de tiempo de interrupciones, lo que indica claramente el efecto negativo de la duración de una interrupción en la QoE del usuario, observando que el usuario tendría que soportar un tiempo de interrupción casi igual al tiempo de visualización del video en tiempo real.

Prueba	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b ∑(Tdatai)	c ∑(Tinti)	Texec Tstart + b
1	2,70	20,30	77,70	23,20	80,40	20,60	81,40	293,30	64,10	239,50	599,60
2	3,00	20,60	80,40	21,10	80,50	20,50	80,26	293,20	62,20	241,16	599,56
3	2,70	20,90	80,50	20,80	80,70	20,30	82,10	291,70	62,00	243,30	599,70

Figura 5.20 Experimentación real

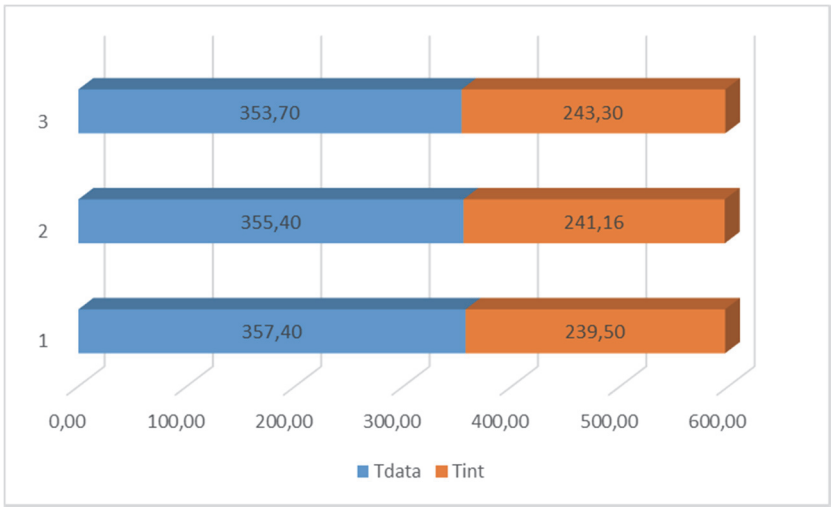

Figura 5.21 Análisis del tiempo de ejecución y de interrupción

Cuando ocurre una interrupción es importante analizar la cantidad de memoria utilizada para grabar los videos que no han podido ser reproducidos en el dispositivo cliente. La Figura 5.22 indica el almacenamiento utilizado durante las 3 interrupciones dadas; el promedio de uso de memoria de cada prueba se muestra en las Figuras 5.23 y 5.24, definiéndose como promedio el uso de 9,20 KB de grabación de video en una interrupción dada. Al realizar el análisis costo beneficio, el costo que involucra tener un almacenamiento temporal se paga con creces al proveer al usuario de servicios adicionales que mejoran la QoE, ya que tiene la opción de visualizar posteriormente esta información almacenada.

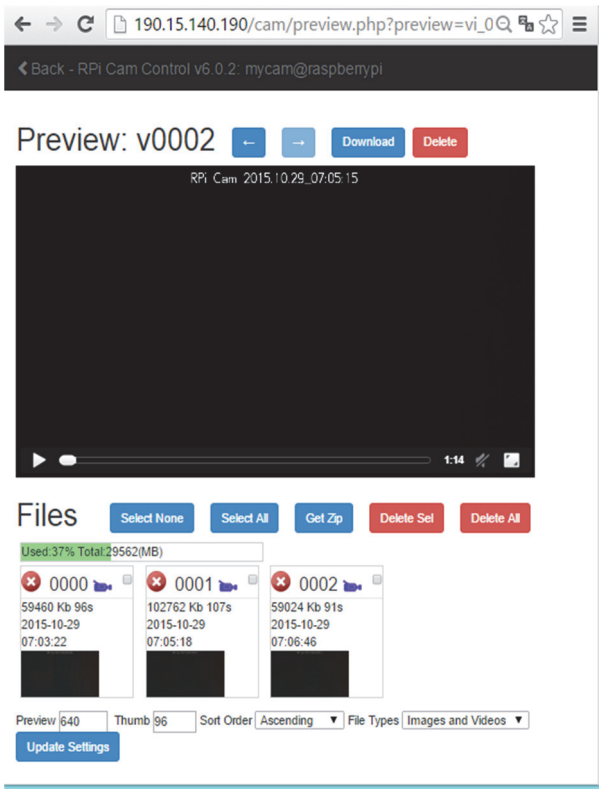

Figura 5.22 Uso de memoria de videos

Promedio Uso Memoria	Promedio duracion video	Uso Memoria KB
73748,67	98,00	9,22
85126,00	141,33	10,64
62033,67	94,00	7,75
73636,11	**111,11**	**9,20**

Figura 5.23 Promedios uso de memoria

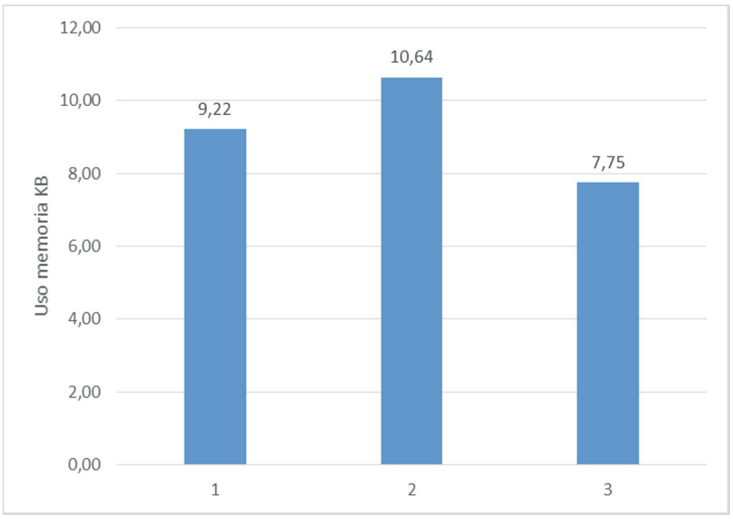

Figura 5.24 Uso de memoria promedio

El segundo escenario donde se realizaron las pruebas en vivo es en la Universidad de las Fuerzas Armadas ESPE, en Ecuador con la duración de 10 minutos, donde los datos de transmisión e interrupción se muestran en la Figura 5.25.

Se puede denotar claramente en la Figura 5.26 que el tiempo de interrupción afecta la trasmisión del video streaming, durante los tiempos de interrupción lo que captura la cámara es grabado en un almacenamiento temporal.

Se ha propuesto una arquitectura denominada ICARTS, que permite la recuperación y continuidad de la transmisión del video streaming en tiempo real en dispositivos inalámbricos en ambientes inalámbricos cuando ocurre una interrupción.

Esta arquitectura, establece ventajas como son la interoperabilidad y multiplataforma al permitir la aplicación de agentes inteligentes como servicios Web. Los agentes desarrollados, presentan un comportamiento ontológico inteligente al analizar constantemente el estado de los señores del ambiente domótico controlado y definir una alerta, activando proactivamente la transmisión del video en tiempo real al cliente. ICARTS, soluciona la presencia de una interrupción mediante un bucle de reconexión que censa constantemente el canal, una vez que se reanuda la comunicación, éste bucle solicita el servicio del video streaming, manteniendo la continuidad de la reproducción.

Prueba	Tstart	Tdata1	Tint1	Tdata2	Tint2	Tdata3	Tint3	Tdatan	b	c	Texec
									∑(Tdatai)	∑(Tinti)	Tstart + b + c + Tdatan
1	2,40	20,60	80,15	20,54	81,03	20,58	80,43	295,17	61,72	241,61	600,90
2	1,90	20,35	80,66	20,58	80,36	20,45	80,50	295,44	61,38	241,52	600,24
3	2,03	20,48	80,56	20,53	80,60	20,61	80,46	295,27	61,62	241,62	600,54

Figura 5.25 Pruebas en vivo de video streaming en tiempo real

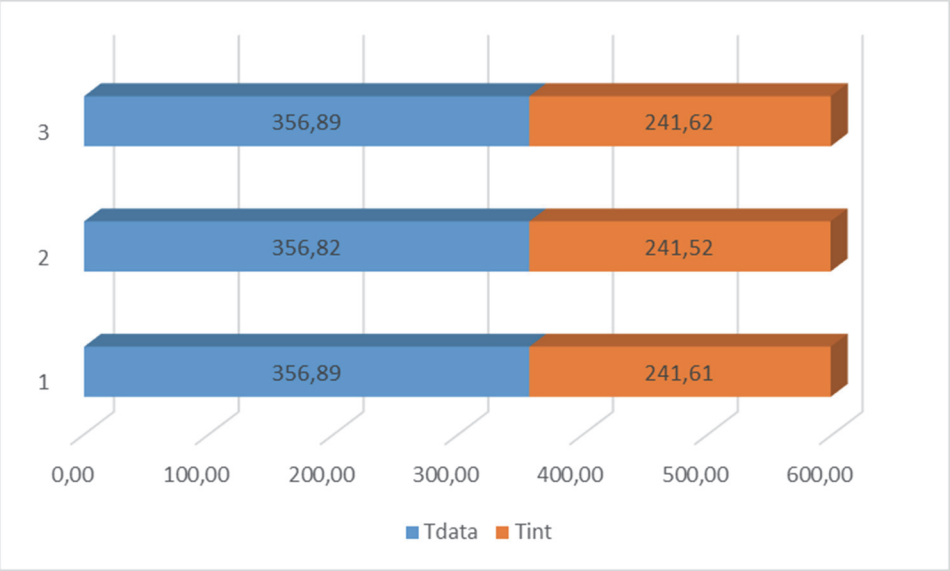

Figura 5.26 Análisis del tiempo de ejecución y de interrupción

Cabe señalar que en la revisión de la literatura no se hallaron referencias a estudios hechos que utilicen transmisión 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

En este capítulo se presentan las conclusiones más importantes de este trabajo de tesis doctoral y las líneas de trabajo futuras.

6.1. Conclusiones

Hemos abordado el problema del servicio de video streaming que al ser transmitido a través de redes WiFi, puede enfrentar varias interrupciones debido a los problemas que pueden darse en la comunicación inalámbrica como latencia, pérdida de paquetes… Este comportamiento impredecible genera interrupciones que ocasionan tiempos adicionales de establecimiento de conexión, degradando la QoE al demandar del usuario una nueva solicitud de transmisión del video streaming desde su inicio. La afectación más importante de estas interrupciones es la cantidad de retransmisiones que ocurren cuyo efecto negativo es el incremento de los tiempos de ejecución en niveles difíciles de tolerar.

Como hemos demostrado se han planteado varias soluciones, en primer lugar una solución básica basado en proxies que define proactivamente una interrupción a través de la observación del canal de comunicación, la cual es solventada mediante el almacenamiento temporal de los frames para que puedan ser posteriormente visualizados al restablecerse la comunicación, se ha logrado eliminar las retransmisiones y tiempos de establecimiento de conexión influyendo favorablemente en el tiempo de ejecución del video y por ende en la QoE del cliente. En segundo lugar, una solución Web interoperable que mediante el análisis ontológico de los recursos del dispositivo móvil define proactivamente una interrupción, este influye positivamente en la QoE al eliminar retransmisiones y demandar menores costos de almacenamiento ya que almacena únicamente información de los recursos y la última posición de reproducción. Finalmente se propone el control de desconexiones en los servicios de video streaming en tiempo real, basados en la información de los sensores se define ontológicamente una alerta de seguridad activando la transmisión en línea, la interrupción es controlada a través de un bucle de reconexión automático que permite mantener la continuidad del servicio frente a una interrupción, logrando buenos niveles de QoE que compensan totalmente el costo del almacenamiento de los videos mientras dura la interrupción.

Como resultado de estas propuestas se ha publicado un artículo en un congreso internacional. Además, 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 profundo de artículos científicos; aplicar técnicas y métodos de ingeniería de software en proyectos eminentemente técnicos fortaleciendo su diseño y arquitectura en base a patrones; a establecer escenarios de experimentación, levantar datos, analizar los resultado e inferir los mismos, también hemos desarrollado la experticia de plantear y ejecutar proyectos de investigación, difundir los resultados en base a artículos científico técnicos. Creemos que todo lo investigado y desarrollado puede tener aplicaciones en servicios multimedia de alta convergencia como IPTv, video conferencia, e-learning, medicina… también en proyectos de computación ubicua, clúster y en la nube; servicios de video streaming basado en sensores para áreas de seguridad, producción, agricultura, desastres naturales… y en particular en nuestro trabajo personal, como docentes investigadores en la constante búsqueda de generar conocimiento y formar investigadores enfocados en plantear ideas innovadoras que busquen el desarrollo y el bienestar social.

Creemos que hemos contribuido a la industria del video streaming porque hemos planteado soluciones que mejoran este servicio considerablemente, al definir propuestas modulares y fáciles de integrar; al proporcionar soluciones acordes al avance tecnológico como es el ambiente Web y uso de dispositivos inalámbricos (móviles, sensores,…), con la finalidad de proporcionar el control de una interrupción, identificándola proactivamente y aplicando estrategias para enfrentarla, logrando aminorar su impacto negativo en la QoE.

6.2. Líneas de trabajo futuro

Fruto de nuestro trabajo se abren líneas de trabajo futuro:

- Web interoperable: un tema interesante es la búsqueda de adicionar mayor versatilidad y otras características a la arquitectura propuesta, transformando funcionalidades y analizar el uso de servidores de video peer to peer como WebRTC para el control de interrupciones.
- Servicios domóticos: el área de automatización de ambientes controlados por hardware libre, es un campo de acción que brinda varias posibilidades

conjuntamente con otras áreas de conocimiento como por ejemplo, en el campo agrícola se podría controlar los cultivos críticos y definir patrones de actuación para mejorar su producción, en el campo de la biotecnología con la información de crecimiento ideal de un cierta especie, se podría recrear ambientes ideales en cualquier ubicación del planeta sin afectaciones externas del medio ambiente, a nivel industrial se pueden prevenir accidentes por emanaciones de gases tóxicos por medio de sensores que oportunamente identifiquen un comportamiento inadecuado, en un data center para el control emergente de contingencias en cuestiones de cortes de energía…

- Servicios de multimedia: nuestro esquema se puede aplicar a transmisiones de video streaming en tiempo real o bajo demanda, puede ser útil en servicios de video conferencia, televisión digital, capacitación en línea….

Glosario

AAC	Advanced Audio Coding
ACC	Agent Communication Channel
ACL	Agent Communication Language
ADSL	Asymmetric Digital Subscriber Line
AMS	Agent Management System
APC	Agent Proxy Client
APS	Agent Proxy Server
BR	Buffering Radio
CCS3	Cascading Style Sheets
CODEC	Compresor-Descompresor
CSELT	Centro Studi e Laboratori Telecommunicazioni
DF	Directory Facilitator
EDGE	Enhaced Data Rates

FIPA	Foundation for intelligent Physical Agent
GPIO	General Purpose Input/Output
GPS	Global Positioning System
HDMI	High Definition Multimedia Interface
HTTP	HyperText Transfer Protocol
ICARTS	Interrupt Control Architecture of Real Time Video streaming
ICASWA	Interrupt Control Architecture of Video streaming by Web Agents
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IOS	iPhone Operating System
IPTV	Internet Protocol Television
ISO/IEC	International Organization for Standarization / International Electrotechnical Commission
ITS	Institute for Telecommunication Science
ITU	International Telecommunication Union

JADE	Java Agent Development Framework
LEAP	Lightweight Extensible Agent Platform
LBR	Low Bitrate Ratio
LED	Light Emitting Diode
LGPL	GNU Library General Public License
LTE	Long Term Evolution
MAS	Multiagent System
MJPEG	Motion Joint Photographic Expert Group
MJPEG	Motion Joint Photographic Expert Group
MOS	Mean Opinion Score
MPQM	Moving Pictures Quality Metric
MTP	Message Transport Protocol
MVC	Model-View-Controller
NFC	Near Field Communication
NQM	Noise Quality Measure

PSNR	Peak-Signal-to-Noise-Ratio
QoE	Quality of Experience
QoS	Quality of Service
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RTP/RTCP	Real Time Transport Protocol/ Real Time Control Protocol
RTSP	Real Time Video Streaming Protocol
SD	Secure Data
SDP	Session Description Protocol
SOAP	Simple Object Access Protocol
SSIM	Structural Similarity Index
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
VoIP	Voice Over Internet Protocol

VQM	Video Quality Metric
WC3	World Wide Web Consortium
WiFi	Wireless Fidelity
WSDL	Web Services Description Language
WSIG	Web Services Integration Gateway
WSN	Wireless Sensor Network
XML	eXtensibility Markup Language

Referencias bibliográficas

[1] J. West and M. Mace, “Browsing as the killer app: Explaining the rapid success of Apple’s iPhone,” *Telecommunications Policy*, vol. 34, no. 5, pp. 270–286, 2010.

[2] 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.

[3] F. Bergenti and A. Poggi, “Leap: A fipa platform for handheld and mobile devices,” in *Intelligent agents VIII*, Springer, 2002, pp. 436–446.

[4] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, “Quantification of YouTube QoE via crowdsourcing,” in *Multimedia (ISM), 2011 IEEE International Symposium on*, 2011, pp. 494–499.

[5] A. Rao, A. Legout, Y. Lim, D. Towsley, C. Barakat, and W. Dabbous, “Network characteristics of video streaming traffic,” in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, 2011, p. 25.

[6] J. Steshenko, V. G. Chaganti, and J. Kurose, “Mobility in a large-scale WiFi network: from syslog events to mobile user sessions,” in *Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, 2014, pp. 331–334.

[7] H. A. Campbell and A. C. La Pastina, “How the iPhone became divine: new media, religion and the intertextual circulation of meaning,” *New Media & Society*, vol. 12, no. 7, pp. 1191–1207, 2010.

[8] F. L. Hernández, *Objective-C. Curso práctico para programadores Mac OS X, iPhone y iPad*. Rc Libros, 2012.

[9] Z. Li, “Application of MVC pattern in data middleware,” *Computer Engineering*, vol. 36, no. 9, pp. 70–72, 2010.

[10] D. Greenwood, P. Buhler, and A. Reitbauer, “Web service discovery and composition using the web service integration gateway,” in *e-Technology, e-Commerce and e-Service, 2005. EEE’05. Proceedings. The 2005 IEEE International Conference on*, 2005, pp. 789–790.

[11] Cisco, “VNI Mobile Forecast Highlights, 2014 – 2019.” http://www.cisco.com/assets/sol/sp/vni/forecast_highlights_mobile/index.html.

[12] Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2014-2019 White Paper.” http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html.

[13] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper." http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[14] M. Zargari, "The path towards Gb/s wireless LANs," in *2012 IEEE Radio Frequency Integrated Circuits Symposium*, 2012.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] J.-H. Lee, K. Morioka, N. Ando, and H. Hashimoto, "Cooperation of distributed intelligent sensors in intelligent environment," *Mechatronics, IEEE/ASME Transactions on*, vol. 9, no. 3, pp. 535–543, 2004.

[24] C. Trasvina-Moreno, A. Asensio, R. Casas, R. Blasco, A. Marco, and others, "Wifi sensor networks: A study of energy consumption," in *Multi-Conference on Systems, Signals & Devices (SSD), 2014 11th International*, 2014, pp. 1–6.

[25] Z. Xingming and Z. Shaoxin, "One load balancing solution for mobile video surveillance system," in *Computer Science & Service System (CSSS), 2012 International Conference on*, 2012, pp. 757–760.

[26] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, "Sensor search techniques for sensing as a service architecture for the internet of things," *Sensors Journal, IEEE*, vol. 14, no. 2, pp. 406–420, 2014.

[27] S. Lee, N. Park, and D. Choi, "Security Management System for Sensor Network," in *International Conference on Convergence and Hybrid Information Technology 2008*, 2008, pp. 806–810.

[28] M. Eid, R. Liscano, and A. El Saddik, "A novel ontology for sensor networks data," in *Computational Intelligence for Measurement Systems and Applications, Proceedings of 2006 IEEE International Conference on*, 2006, pp. 75–79.

[29] M. Eid, R. Liscano, and A. El Saddik, "A universal ontology for sensor networks data," in *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on*, 2007, pp. 59–62.

[30] F. Abdoli and M. Kahani, "Ontology-based distributed intrusion detection system," in *Computer Conference, 2009. CSICC 2009. 14th International CSI*, 2009, pp. 65–70.

[31] L. Li, G. Xing, L. Sun, and Y. Liu, "A Quality-Aware Voice Streaming System for Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 4, p. 61, 2014.

[32] E. Karapistoli, A. Economides, and others, "Wireless sensor network security visualization," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on*, 2012, pp. 850–856.

[33] K. Miller and S. Suresh, "Monitoring patient health using policy based agents in wireless body sensor mesh networks," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 2009, pp. 503–508.

[34] C. Bell, *Beginning sensor networks with Arduino and Raspberry Pi*. Apress, 2013.

[35] R. Components, "La conectividad de las redes móviles en Raspberry Pi y Arduino," *Revista española de electrónica*, no. 714, pp. 46–47, 2014.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] M. Xing, S. Xiang, and L. Cai, "A real-time adaptive algorithm for video streaming over multiple wireless access networks," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 4, pp. 795–805, 2014.

[41] P. Van Beek and L. J. Kerofsky, "Server-Side Playout Delay Management for Video Streaming," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 3077–3080.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] J. Lou, S. Liu, A. Vetro, and M. T. Sun, "Complexity and memory efficient GOP structures supporting VCR functionalities in H. 264/AVC," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, 2008, pp. 636–639.

[52] 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.

[53] Y. Reznik, E. Asbun, Z. Chen, Y. Ye, E. Zeira, R. Vanam, Z. Yuan, G. Sternberg, A. Zeira, and N. Soni, "User-adaptive mobile video streaming," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*, 2012, pp. 1–1.

[54] M. Thejaswini, P. Rajalakshmi, and U. B. Desai, "Novel Sampling Algorithm for Human Mobility Based Mobile Phone Sensing."

[55] Ooyala, "Global video Index Q1 2015." http://go.ooyala.com/rs/OOYALA/images/Ooyala-Global-Video-Index-Q1-2015.pdf.

[56] N. Staelens, J. De Meulenaere, M. Claeys, G. Van Wallendael, W. Van den Broeck, J. De Cock, R. Van de Walle, P. Demeester, and F. De Turck, "Subjective quality assessment of longer duration video sequences delivered over HTTP adaptive streaming to tablet devices," *Broadcasting, IEEE Transactions on*, vol. 60, no. 4, pp. 707–714, 2014.

[57] 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.

[58] 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.

[59] Y. A. P. Leal and L. G. M. Ballesteros, "Factores que Influyen en la Evaluación de QoE del Servicio de Video Streaming."

[60] 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.

[61] 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.

[62] 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.

[63] A. Iqbal, F. Arif, and N. Minallah, "Performance evaluation of stack-protocols, encapsulation methods and video codecs for live video streaming," in *Information and Communication Technology (ICoICT), 2013 International Conference of*, 2013, pp. 223–228.

[64] 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.

[65] 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.

[66] 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.

[67] Y. Wang, "Survey of objective video quality measurements," 2006.

[68] 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.

[69] 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.

[70] 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.

[71] H. Kowshik, P. Dutta, M. Chetlur, and S. Kalyanaraman, "A quantitative framework for guaranteeing QoE of video delivery over wireless," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 290–294.

[72] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[73] P. Pérez, J. Ruiz, and N. Garcıa, “Calidad de Experiencia en servicios multimedia sobre IP,” *Jornadas Telecom I+ D (Telecom I+ D 2010)*, 2010.

[74] A. Khan, L. Sun, and E. Ifeachor, “Impact of video content on video quality for video over wireless networks,” in *Autonomic and Autonomous Systems, 2009. ICAS’09. Fifth International Conference on*, 2009, pp. 277–282.

[75] S. Abdallah-Saleh, Q. Wang, C. Grecos, and D. Thomson, “Handover evaluation for mobile video streaming in heterogeneous wireless networks,” in *Electrotechnical Conference (MELECON), 2012 16th IEEE Mediterranean*, 2012, pp. 23–26.

[76] 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.

[77] I. Iskandar and Y. Bandung, “Supporting QoS in multimedia over 802.11 e wireless network with adaptive framework,” in *Cloud Computing and Social Networking (ICCCSN), 2012 International Conference on*, 2012, pp. 1–4.

[78] S. Misra, S. N. Das, and M. S. Obaidat, “Context-Aware quality of service in wireless sensor networks,” *Communications Magazine, IEEE*, vol. 52, no. 6, pp. 16–23, 2014.

[79] P. O. Flaithearta, H. Melvin, and M. Schukat, “A QoS enabled WiFi AP,” in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, 2014, pp. 1–4.

[80] H. Zhu, M. Li, I. Chlamtac, and B. Prabhakaran, “A survey of quality of service in IEEE 802.11 networks,” *Wireless Communications, IEEE*, vol. 11, no. 4, pp. 6–14, 2004.

[81] B. Amin and F. Didi, “QoS: from WIFI to UMTS,” in *Innovation Management and Technology Research (ICIMTR), 2012 International Conference on*, 2012, pp. 193–198.

[82] S. Paramasivam and K. Thyagharajan, “Comparative study and performance evaluation of multimedia scheduling in wireless networks,” in *Green Computing, Communication and Conservation of Energy (ICGCE), 2013 International Conference on*, 2013, pp. 209–214.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] 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.

[88] 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.

[89] 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.

[90] D. Kumar, S. A. Srinivas, P. Hariharan, and R. A. Kumar, “Cross-layered implementation of multilayered coded video streaming,” in *Recent Trends in Information Technology (ICRTIT), 2013 International Conference on*, 2013, pp. 680–684.

[91] J. Shen, B. Han, M.-C. Yuen, and W. Jia, “End-to-end wireless multimedia transmission system,” in *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, 2004, vol. 4, pp. 2616–2620.

[92] J.-W. Ding, C.-T. Lin, and K.-H. Huang, “ARS: an adaptive reception scheme for handheld devices supporting mobile video streaming services,” in *Consumer Electronics, 2006. ICCE’06. 2006 Digest of Technical Papers. International Conference on*, 2006, pp. 141–142.

[93] 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.

[94] 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.

[95] 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.

[96] X. Wang, M. Chen, T. T. Kwon, L. Yang, and V. Leung, "AMES-Cloud: a framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *Multimedia, IEEE Transactions on*, vol. 15, no. 4, pp. 811–820, 2013.

[97] F. Molazem Tabrizi, J. Peters, and M. Hefeeda, "Dynamic control of receiver buffers in mobile video streaming systems," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 5, pp. 995–1008, 2013.

[98] S. Ickin, M. Fiedler, and K. Wac, "Demonstrating the stalling events with instantaneous total power consumption in smartphone-based live video streaming," in *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*, 2012, pp. 1–4.

[99] 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.

[100] 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.

[101] 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.

[102] S. Bogle and S. Sankaranarayanan, "Intelligent agent based job search system in Android environment," in *Electro/Information Technology (EIT), 2011 IEEE International Conference on*, 2011, pp. 1–6.

[103] M. Wooldridge, *An Introduction to MultiAgent Systems 2nd*. Wiley Publishing, 2009.

[104] S. J. R. Peter Norvig, *Inteligencia Artificial Un Enfoque Moderno*. Prentice Hall Hispanoamericana, 1996.

[105] A. Diosteanu, L. Cotfas, A. Smeureanu, and S. D. Dumitrescu, "Multi-agents and GIS framework for collaborative supply chain management applications," in *Roedunet International Conference (RoEduNet), 2010 9th*, 2010, pp. 157–162.

[106] C. McTavish and S. Sankaranarayanan, "Intelligent agent based hotel search & booking system," in *Electro/Information Technology (EIT), 2010 IEEE International Conference on*, 2010, pp. 1–6.

[107] R. C. Moore, "A formal theory of knowledge and action," Stanford Research Institute (Menlo Park, CA US), AI-TN-320, 1984.

[108] M. Wooldridge and P. Ciancarini, "Agent-oriented Software Engineering: The State of the Art," in *First International Workshop, AOSE 2000 on Agent-oriented Software Engineering*, 2001, pp. 1–28.

[109] S. DeLoach and M. F. Wood, "Developing Multiagent Systems with agentTool," in *Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*, 2001, pp. 46–60.

[110] G. M. Rajakaruna, A. S. Karunananda, and S. Jayalal, "Aligning ontologies using Multi Agent technology," in *Advances in ICT for Emerging Regions (ICTer), 2012 International Conference on*, 2012, pp. 189–196.

[111] T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *Int. J. Hum.-Comput. Stud.*, vol. 43, no. 5–6, pp. 907–928, 1995.

[112] N. Guarino, "Formal Ontology and Information Systems," 1998, pp. 3–15.

[113] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering: Principles and Methods," *Data Knowl. Eng.*, vol. 25, no. 1–2, pp. 161–197, 1998.

[114] J. Yi, B. Ji, C. Chen, and X. Tian, "Employing Ontology to build the engine fault diagnosis expert system," in *Computer Science and Information Engineering, 2009 WRI World Congress on*, 2009, vol. 2, pp. 631–635.

[115] F. I. Hernandez, C. A. Canesin, R. Zamora, and A. K. Srivastava, "Active power management in multiple microgrids using a multi-agent system with JADE," in *Industry Applications (INDUSCON), 2014 11th IEEE/IAS International Conference on*, 2014, pp. 1–8.

[116] P. Balaji and D. Srinivasan, "Multi-agent system in urban traffic signal control," *Computational Intelligence Magazine, IEEE*, vol. 5, no. 4, pp. 43–51, 2010.

[117] B. M. Balachandran and M. Enkhsaikhan, "Development of a multi-agent system for travel industry support," in *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, 2006, pp. 63–63.

[118] L. Gloss, B. Mason, and J. McDowall, "Multi-agent System Service and Ontology Design Methods," in *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, 2008, pp. 453–456.

[119] G. Nguyen, T. T. Dang, L. Hluchy, Z. Balogh, M. Laclavik, and I. Budinska, "Agent platform evaluation and comparison," *Rapport technique, Institute of Informatics, Bratislava, Slovakia*, 2002.

[120] M. Naghibzadeh and A. Rahimi, "Communication protocols for distributed reasoning in expert systems," in *Advanced Computational Intelligence (IWACI), 2010 Third International Workshop on*, 2010, pp. 511–516.

[121] S. Sotiriadis, N. Bessis, Y. Huang, P. Kuonnen, and N. Antonopoulos, "A JADE Middleware for Grid inter-cooperated Infrastructures," in *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, 2011, pp. 135–140.

[122] N. T. M. Khue, "Developing an Intelligent Multi-Agent System based on JADE to solve problems automatically," in *Systems and Informatics (ICSAI), 2012 International Conference on*, 2012, pp. 684–690.

[123] C. Xiong, W. Xiang, and Y. Ouyang, "Argumentation in multi-agent system based on JADE," in *Intelligent Control and Information Processing (ICICIP), 2012 Third International Conference on*, 2012, pp. 88–91.

[124] X. Jinkai and Y. Weihong, "Study on comparison between JAFMAS and JADE," in *Circuits, Communications and System (PACCS), 2010 Second Pacific-Asia Conference on*, 2010, vol. 1, pp. 105–108.

[125] L. Yu and Z. Mbumwae, "Towards a multi-agent system (MAS) based Credit Reference Bureau," in *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, 2009, vol. 1, pp. 728–732.

[126] D. Meere, I. Ganchev, M. O'Dróma, S. Stojanov, and V. Valkanova, "Adaptation for Assimilation: Shaping Context-Sensitive M-Learning Services within a Multi-Agent Environment," in *Telecommunications (AICT), 2010 Sixth Advanced International Conference on*, 2010, pp. 74–79.

[127] R. A. Brown and S. Sankaranarayanan, "Intelligent agent based mobile shopper," in *Wireless and Optical Communications Networks, 2009. WOCN'09. IFIP International Conference on*, 2009, pp. 1–7.

[128] W. Ma and D. H. Du, "Reducing bandwidth requirement for delivering video over wide area networks with proxy server," *Multimedia, IEEE Transactions on*, vol. 4, no. 4, pp. 539–550, 2002.

[129] W. Ma and D. H. Du, "Design a progressive video caching policy for video proxy servers," *Multimedia, IEEE Transactions on*, vol. 6, no. 4, pp. 599–610, 2004.

[130] 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.

[131] G. Wu, E. K. Chong, and R. Givan, "Streaming stored video over AIMD transport protocols," in *Multimedia Software Engineering, 2002. Proceedings. Fourth International Symposium on*, 2002, pp. 304–311.

[132] 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.

[133] 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.

[134] 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.

[135] J. Choi, J. Kim, and B. Jang, "A Software Wireless Streaming Architecture Supporting telematics device," in *2007 Digest of Technical Papers International Conference on Consumer Electronics*, 2007.

[136] 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.

[137] 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.

[138] 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.

[139] 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.

[140] 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.

[141] 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.

[142] 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.

[143] 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.

[144] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[145] C. Xu, P. Liang, T. Wang, Q. Wang, and P.-Y. Sheu, "Semantic web services annotation and composition based on er model," in *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, 2010, pp. 413–420.

[146] D. Sachan, S. K. Dixit, and S. Kumar, "A system for Web Service selection based on QoS," in *Information Systems and Computer Networks (ISCON), 2013 International Conference on*, 2013, pp. 139–144.

[147] K. S. Wagh and R. C. Thool, "Performance analysis of mobile web service provisioning on different mobile host," in *India Conference (INDICON), 2014 Annual IEEE*, 2014, pp. 1–5.

[148] M. Ali, H. Sun, and W. Yuan, "An Efficient Routing Scheme for Overlay Network of SOAP Proxies in Constrained Networks," in *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*, 2013, pp. 466–473.

[149] A. Al-Rokabi and C. Politis, "SOAP: A cognitive hybrid routing protocol for Mobile Ad-Hoc Networks," in *Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM), 2014 9th International Conference on*, 2014, pp. 353–359.

[150] F. AlShahwan and K. Moessner, "Providing soap web services and restful web services from mobile hosts," in *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, 2010, pp. 174–179.

[151] Y. Xu, S. Ma, S. Yi, and Y. Yan, "From XML Schema to Relations: A Incremental Approach to XML Storage," in *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, 2010, pp. 1–4.

[152] K. Aljoumaa, S. Assar, and C. Souveyet, "Publishing Intentional Services using extended semantic annotation," in *Research Challenges in Information Science (RCIS), 2011 Fifth International Conference on*, 2011, pp. 1–9.

[153] L. Xin, Z. Feng, and S. Chen, "Experimental Study of Web Service Availability Based on WSDL Documents," in *Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on*, 2013, pp. 356–362.

[154] M. Sarith Divakar and K. Mathew, "UDDI Rated Web Services," in *Advances in Computing and Communications (ICACC), 2013 Third International Conference on*, 2013, pp. 340–342.

[155] C.-H. Liang, W.-S. Hung, M.-C. Hsieh, C.-M. Wu, and C.-H. Luo, "A multi-agent based architecture for an assistive user interface of intelligent home environment control," in *Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on*, 2008, vol. 1, pp. 335–338.

[156] G. Sharath, "JavaScript: The Used Parts," 2014.

[157] H. Prima Dewi Purnamasari and N. Syifana, "Clickable and interactive video system using HTML5," in *Information Networking (ICOIN), 2014 International Conference on*, 2014, pp. 232–237.

[158] H. V. Nguyen, H. A. Nguyen, T. T. Nguyen, A. T. Nguyen, and T. N. Nguyen, "Dangling references in multi-configuration and dynamic PHP-based Web applications," in *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, 2013, pp. 399–409.

[159] S. Pasqualini, F. Fioretti, A. Andreoli, and P. Pierleoni, "Comparison of H. 264/AVC, H. 264 with AIF, and AVS based on different video quality metrics," in *Telecommunications, 2009. ICT'09. International Conference on*, 2009, pp. 190–195.

[160] S. Paulikas, "Estimation of video quality of H. 264/AVC video streaming," in *EUROCON, 2013 IEEE*, 2013, pp. 694–700.

[161] Y.-S. Li, C.-C. Chen, T.-A. Lin, C.-H. Hsu, Y. Wang, and X. Liu, "An end-to-end testbed for scalable video streaming to mobile devices over HTTP," in *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, 2013, pp. 1–6.